DEC-6-0-EX-SYS-UM-FP-ACT00

MULTIPROGRAMMING SYSTEM MANUAL

PDP-6

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

DEC-6-0-EX-SYS-UM-FP-ACT00

PDP-6 MULTIPROGRAMMING SYSTEM MANUAL

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

PREFACE

This manual describes how to use the Multiprogramming Monitor version 1. The distinguishing feature of this version is that all users' programs together must fit within the limits of core memory and are prevented from interfering with each other by hardware and with each other's I/O by software. Also, DECtape is the principal storage medium for users' files.

As future versions of the monitor are developed, inevitably corresponding changes will be made in existing users' programs, both to take advantage of newly added features and to make corrections where an old feature was changed. Therefore, it is recommended that the monitor- and I/O-interface portion of the user's program be written as a separate program so that it may be updated easily. Every effort will be made to keep such changes to a minimum.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

ILLUSTRATIONS

TABLES

CHAPTER 1

INTRODUCTION TO THE PDP-6 MULTIPROGRAMMED TIME-SHARING SYSTEM

The Programmed Data Processor-6 time-sharing system provides integrated software and hardware for the simultaneous use of a single system by many users at Teletype consoles. A user may obtain the use of all system facilities and programs from a remote location. Users' programs and data are kept on a separate DECtape for each user to permit permanent storage of results.

A user may start one or more active programs called jobs. Each job is console initiated and runs in an assigned and protected memory area.

BASIC SYSTEM FEATURES

The basic system objective is to provide a simple, modular, time-sharing system.

File Generation - A user may create and edit a text file directly from his time-sharing console. The need for off-line program preparation devices (key punches, paper tape perforators) is greatly reduced.

On-Line Debugging - A user may debug his program on-line through an advanced debugging program (DDT). Corrections can be made and tested instantly. On-line debugging is feasible because a user cannot tie up the entire installation while he stops to decide what to do next.

Common User Service Programs (CUSPS) - Commonly used programs, such as the compiler, the assembler, and file manipulation programs, are available to all users. New programs of general interest are easily made available to all users.

Unattended Batch Processing - The user may call for a special user program called the Batch Control Processor (BCP) which then takes over the supervision of his job in a manner similar to conventional batch processing systems.

Special Console Service - The executive subroutines may be modified so that installations requiring a special service or language may be included.

Real Time Processing - The necessary input/output routines to connect a special device with the programming system can be added. A job could then issue system input/output commands for the special device in a similar manner as for conventional devices. In some applications, it would be necessary to modify the schedular portion of the monitor so that occurrence of I/O device conditions would cause special service for the user.

Common I/O Service - I/O service routines within the monitor provide a basic set of input/output commands. Programs are easily written which communicate with many I/O devices using identical commands for each device. All I/O subroutines use the hardware priority interrupt system so that the monitor never waits for data transmission.

Common System Service - Monitor routines, at the request of a user program, perform special system functions such as giving the time, date, and job reinitializing.

HARDWARE REQUIREMENTS

The following is the minimum hardware configuration to allow multiprogrammed time sharing.

1-166	Arithmetic Processor with 626 Teleprinter
1-161C (or 163C)	16384-Word Core Memory
1-136	Data Control
1-551	DECtape Control
1-555	Dual DECtape Transport (2 drives)
1-760	Paper Tape Reader (for maintenance programs)

In addition to the above hardware, one Type 162 16-Word Fast Memory and any number of additional Type 161C (or 163C) Core Memories (up to a total of 262,144 words) may be added to the system without requiring any modifications to system software.

Optional Hardware with Software Support

With very minor modifications to system software, the standard Time-Sharing Monitor can handle any configuration that contains the following devices:

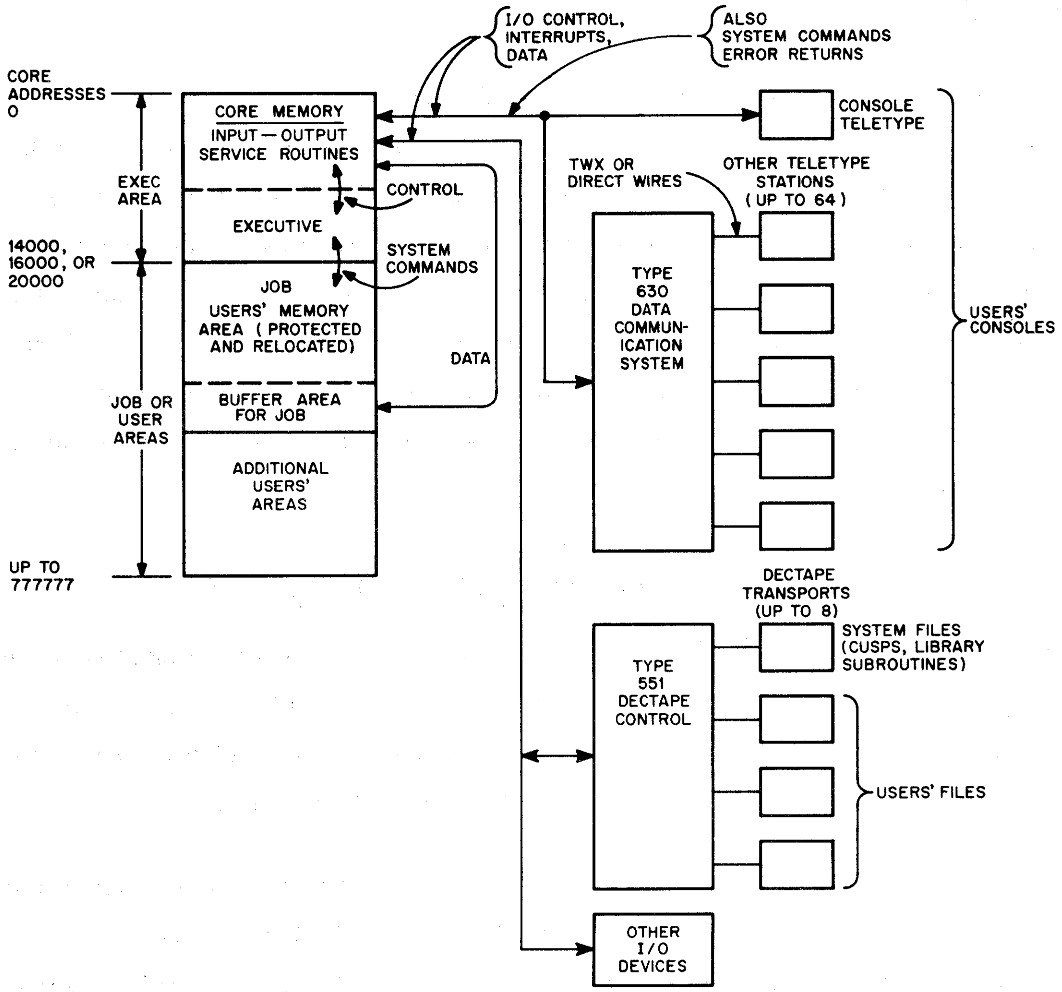
1-630	Data Communications System with up to 64 half- or full-duplex 8-level teletypewriters
4-555	Dual DECtape Transports (up to 8 drives)
1-516	Magnetic Tape Control with up to 8 Magnetic Tape Transports Type 50, Type 570, or IBM Type 729 (any model)

1-346	Incremental Display (with optional light pen and character generator)
1-461	Card Reader (200 or 800 cpm)
1-646	Line Printer (300, 600 or 1000 lpm)
1-761	Paper Tape Punch

CHAPTER 2

MULTIPROGRAMMING SYSTEM BLOCK DIAGRAM DISCUSSION

Figure 2-1, the Multiprogramming System Block Diagram, depicts the flow of data to the user consoles and external files, and the software-hardware interaction.



I/O CONTROL, INTERRUPTS, DATA
ALSO SYSTEM COMMANDS ERROR RETURNS
CORE ADDRESSES 0
EXEC AREA
14000, 16000, OR 20000
JOB OR USER AREAS
UP TO 777777
CORE MEMORY
INPUT—OUTPUT SERVICE ROUTINES
EXECUTIVE
JOB
USERS' MEMORY AREA (PROTECTED AND RELOCATED)
BUFFER AREA FOR JOB
ADDITIONAL USERS' AREAS
CONTROL
SYSTEM COMMANDS
DATA
CONSOLE TELETYPE
TWX OR DIRECT WIRES
OTHER TELETYPE STATIONS (UP TO 64)
TYPE 630 DATA COMMUN-ICATION SYSTEM
USERS' CONSOLES
DECTAPE TRANSPORTS (UP TO 8)
SYSTEM FILES (CUSPS, LIBRARY SUBROUTINES)
TYPE 551 DECTAPE CONTROL
USERS' FILES
OTHER I/O DEVICES

Figure 2-1 Multiprogramming System Block Diagram

USER CONSOLES

As shown in the system block diagram, a user's console is either the console Teletype at the computer or one of 64 Teletypes at (possibly) remote locations connected to the computer via the data communications system. The console is a user's communication with the multiprogramming system and with his files. All the commands necessary for starting and controlling a job are made via this console; system responses and error messages are printed on the user's console. Also, the console is useful for entering data and receiving results.

Teletype stations connect to the data communications system through standard TWX lines or direct wire. Characters are processed sequentially as they arrive according to one of several schemes as selected by the operating program. Characters are transmitted to the user's program a character at a time, a line at a time, or a full buffer (95 characters) at a time.

A user's console has three states: the detached state, the monitor state, and the user state. The detached state exists whenever the console is idle. If a user types a legitimate monitor command on the detached console, the monitor automatically assigns a free job number to the console and places the console in the monitor state. In this state, the monitor interprets characters typed in as monitor commands. Certain monitor commands start the user's program running and simultaneously place the console in the user state so that characters typed in are processed by the user's program and not by the monitor. Certain error conditions or typing the proper control character (C with the control key held down) cause the monitor to return the console to the monitor state. Certain other monitor commands return the console to the detached state (DETACH and KJOB - kill job).

MULTIPROGRAMMING EXECUTIVE FUNCTIONS

The multiprogramming system consists of routines to allocate resources (core storage, arithmetic processor time, I/O devices) among the various users and routines to perform input/output. The resource-allocation portion is called the exec or the monitor. Also included in the exec are routines to perform such bookkeeping operations as timing the user program or supplying the date on request.

The monitor and I/O routines form a collection of programs that are permanently in core memory during time-shared operation. The monitor makes use of the PDP-6 time-sharing hardware to load several users' programs into core memory simultaneously and to prevent each user's program from interfering with any other. If more than one user requests that his program be run, the monitor switches control to each program in a round-robin fashion. Switching is rapid so that all programs appear to run simultaneously. If a user's program must wait for the completion of an input/output operation, the monitor immediately switches to another user so that no computing time is wasted.

INPUT/OUTPUT FUNCTIONS

The I/O service routines relieve the programmer of the burden of writing a unique program for each different I/O device. One set of input/output commands controls any device; to change devices, the program need change only the device name in the initialize command (see INIT, Chapter 4). The I/O service routines perform the more common format translations.

The I/O service routines take full advantage of the program interrupt system of the PDP-6 to allow overlapped computation with input/output. The hardware features of the PDP-6 permit devices with relatively high data rates (high density magnetic tape, for example) to be operated without a separate data channel and without taking up a large percentage of the memory cycles.

The I/O service routines are organized for ease in adding a service routine for a new or unique device. In addition to the new service routine for controlling the new device, only one of the monitor subprograms, a routine called IOINIT, need be modified to inform the system about the new device. Thus, the monitor for a particular installation consists of a number of executive routines that are the same for all installations; a selected set of standard library I/O subroutines to control only those devices actually present at the installation; and one subroutine, unique to the particular installation, that informs the system of which devices are present.

USER'S CORE STORAGE

Figure 2-2 is a simplified diagram of the job storage area showing the normal usage of a user's core memory allotment. The user requests as much core memory space as he requires. Any number of contiguous 1K (1024-word) blocks may be assigned to the user. The first 140 (octal) locations are the job data area and are reserved by the exec for use when providing service for the user. Some system information and temporary storage for accumulators are found in this area. To protect the system information, the exec keeps a private copy while allowing the user's program to run and restores the job data area upon stopping the user's program.

The user's program, data, debugging programs, arrays, etc., are loaded in the user's area starting at location 140. Although the actual location may be anywhere in memory, the program is stored in core as if the first address actually were 140; reference to the proper addresses is made through the hardware relocation system. The memory protection system prevents a user's program from referencing any address outside its own job area and thus presents disturbing another user's program. In addition, hardware prohibits a user's program from trying to halt the processor or influence the I/O or priority interrupt system.

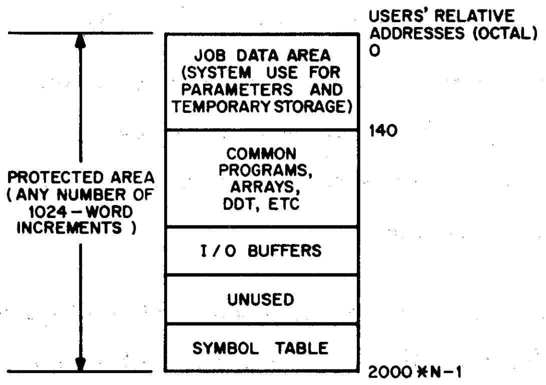

USERS' RELATIVE ADDRESSES (OCTAL)
0
JOB DATA AREA (SYSTEM USE FOR PARAMETERS AND TEMPORARY STORAGE)
140
COMMON PROGRAMS, ARRAYS, DDT, ETC
PROTECTED AREA (ANY NUMBER OF 1024-WORD INCREMENTS)
I/O BUFFERS
UNUSED
SYMBOL TABLE
2000*N-1

Figure 2-2 Simplified User's Core Storage Map

When a user's program performs input/output, data is transmitted directly to or from buffers in the job's own area. Buffers are created dynamically as required. Space for buffers is taken from the unused memory at the end of the user's program; the address of the first unused location is kept in the job data area.

During program loading, the loader builds a symbol table for linking global symbols and for reference by DDT. If DDT is not being used, the symbol table is just additional unused space at run time. If, however, DDT is being used, the user must write his program (or assign enough extra core space) so that the I/O buffers do not build on top of the symbol table.

USER AND SYSTEM FILES

A user's active files are stored on his own reel of DECtape. However, information may originate from, or be sent to, any other device when desired. To call for a file, the user (or the user's program) must specify a device name, a file name with an optional second name called the extension, and (where ambiguous) a data mode. The device name is a 1- to 6-character mnemonic such as PTP for paper tape punch or DTA5 for DECtape transport number 5. Table 4-1 gives a complete list of device names. The file name is any 1- to 6-character name (consisting of letters and numbers) selected by the user at the time the file was written. The name extension is an additional 1- to 3-character mnemonic describing the information contained in the file; for example, REL for a relocatable binary file, MAC for a Macro language source file, LST for an assembly listing file. Typical data modes are ASCII (A) for ASCII data consisting of 7-bit characters packed five to a word, binary (B), dump (D) for core images, and image (I) for unprocessed data received directly from a device.

System files consisting of copies of each of the active CUSP and library programs are kept on one DECtape reel mounted on device SYS (usually DECtape transport 0). The files on this tape are accessible by all users. A listing of the file names of a standard system-files tape (called the CUSP tape) and directions on how to use each of these files appear in Chapter 6.

CHAPTER 3

TIME-SHARING CONSOLE OPERATION

The user has complete access to all system functions and programs by typing on his time-sharing console. User requests are processed in one of two distinctly different ways. Certain requests, such as starting or stopping a program or assigning I/O devices, are performed by the time-sharing monitor itself. Other functions, such as listing a DECtape file, assembling or compiling source language programs, or loading relocatable binary programs, are performed by library programs that are loaded and run in exactly the same manner as any user's program. These library programs are called CUSPs (Common User Service Programs). This chapter explains how to give the commands accepted by the time-sharing monitor; succeeding chapters explain how to give commands to the various programs in the standard CUSP library.

CONSOLE STATES

From the user's point of view, his time-sharing console is in one of three modes (Figure 3-1): the monitor mode, the user mode, or the detached mode. In the monitor mode, characters typed in are presented to the monitor command processor to be interpreted as a monitor command. In the user mode, characters typed in are processed or ignored at the option of the user program. In the user mode the time-sharing console acts as an ordinary input/output device. A time-sharing console is in a third state (the detached state) if nothing has been typed on it since the monitor was started or if the KJOB (kill job) or DETACH command is typed. The console is automatically placed in the monitor mode as soon as any proper monitor command is typed, in which case, the monitor types back:

JOB N INITIALIZED

where N is a number assigned by the monitor for the purpose of identifying this job.

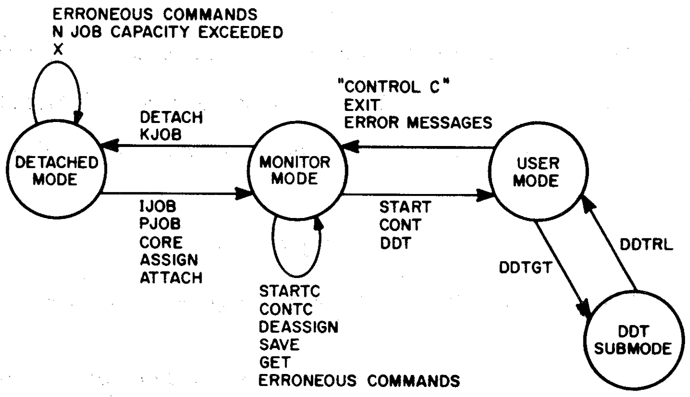


Figure 3-1 Console State Diagram

There are three ways to place the console into the user mode and three ways to return the console to the monitor mode. Typing one of the three monitor commands—START, CONT, and DDT (explained below), places the console in the user mode. Holding down the control key and typing C places the console in the monitor mode and stops the user program if it is running. The other two methods of returning to monitor mode are under program control. The user's program may give the EXIT program command which causes the word EXIT to print on the time-sharing console, the program execution to stop, and the console to enter monitor mode. If the user's program (or the monitor while servicing the user's program) commits a blunder, such as an attempted memory reference outside the user's alloted memory area, the monitor types:

ERROR IN JOB N

where N is the job number assigned by the monitor to the user's program, followed by a 1-line description of the error. After the error printout, the user's program is stopped and the console is placed in monitor mode.

Upon completion of most commands, the monitor types an acknowledgement (usually a carriage-return/line feed) and waits for further commands. If the user types 'control' C before the acknowledgement is typed, the monitor command in progress is stopped before completion.

CONSOLE COMMAND FORMAT

All console commands have the same general format. Command names are strings of from one to six letters and numbers. Characters after the sixth are ignored. Arguments, when required, follow the command name on the same line. Space, and any printing character that is not a letter or numeral, is considered a separator and may be used to separate a command name from its argument(s). Spaces and nonprinting characters preceding the command name are ignored. A semicolon preceding a line causes the line to be disregarded. Other separator characters preceding the line cause the line to be treated as an error. A command line is terminated by carriage-return/line-feed, vertical tab, form-feed, or alt mode. Additional character processing occurs which is the same as for the Teletype ASCII mode described on page 4-18.

If the monitor does not understand the typed-in command, it types back the command up to the place it could not interpret and appends a question mark followed by the words "MONITOR COMMAND ERROR." If the command name is recognized, but a necessary argument for the command is missing, the monitor types back:

TOO FEW ARGUMENTS

If the user types in more arguments than necessary, the extra arguments are ignored.

When a command is typed in correctly, the monitor processes the command without further acknowledgment until either processing is complete, an error occurs in processing the command, or 'control' C is typed. Most commands are processed instantly. The completion of any command causes an extra carriage-return/line-feed or a line of printing followed by carriage-return/line-feed to be typed out.

In periods of heavy use of the time-sharing system, the job capacity of the monitor may be exceeded. The job capacity is limited by table space within the monitor to a number of jobs equal to the number of time-sharing consoles physically connected to the system. If any console initiates more than one job, it is possible to exceed the job capacity of the system. When the job capacity is exceeded, any command typed in to initialize a new job (implicitly as well as explicitly) causes the monitor to respond with:

```
JOB CAPACITY EXCEEDED
```

If this message occurs, the command is not executed and the user must wait until some other user relinquishes his job number (see the explanation of the KJOB command on page 3-9).

When a user detaches his console from his job, a logical console is left attached to the job while the physical console is allowed to attach itself to some other logical console. Therefore, when the number of active jobs is equal to the console-handling capacity of the system, the monitor is unable to find a logical console to which to attach the next physical console that types in. If all logical consoles are in use, a physical console that is not already attached to a logical console cannot communicate with the monitor. To inform the user that the console capacity of the system is exceeded, the monitor types the letter X each time the user types any character other than X. To make this problem as infrequent as possible, the monitor console capacity is one more than the job capacity. Thus, when the job capacity is used up, the next user to type receives the "JOB CAPACITY EXCEEDED" message via the one remaining logical console; but, if yet another user starts typing, he receives the "X" indication. However, if a console receives the "JOB CAPACITY EXCEEDED" message, the monitor breaks the connection to the logical console automatically at the end of the message, thus freeign the logical console for the use of anyone who might be receiving the "X" message.

> NOTE: When the "JOB CAPACITY EXCEEDED" message is received, the only console command that has a useful effect is ATTACH (see the explanation of ATTACH on page 3-10).

Console Commands for Initializing a Job

The following commands are most often typed by the user when he first sits down at the console.

IJOB — Initialize JOB. (Required only if someone else has been using the same console.) This command initializes the job to the following state: no core assigned; no devices except the Teletype assigned; the Teletype given the logical name TTY. The monitor responds with:

```
JOB N INITIALIZED
```

PJOB	Print JOB number. The monitor types out the number assigned to the job to which the console is currently attached.

ASSIGN ASSIGN I/O devices to this job. This command requires one or two arguments. In the form:

ASSIGN DEV

the physical device DEV is reserved for the exclusive use of this job. (See Table 4-1 for a complete list of devices and their physical names.)

In the form:

ASSIGN DEV NAME

device DEV is reserved, and in addition, NAME (called the logical name) is made synonymous with DEV. If NAME happens to be the same as the physical name of some other device, the logical name takes precedence. With the ability to rename devices, a user may write his programs to use arbitrarily named devices which he assigns to the most convenient physical devices at run time.

For multiple devices such as DECtape or magnetic tape, the physical name consists of a 3-letter device-type mnemonic followed by the unit (transport) number. For example, DTA3 means DECtape (DTA) transport number 3. To locate a free transport, use only the first three characters of the physical device name in place of DEV. For example:

```
User types:          ASSIGN DTA
Monitor responds:    DEVICE DTA4 ASSIGNED
Then user types:     ASSIGN DTA
Monitor responds:    DEVICE DTA5 ASSIGNED
```

In the example above, the monitor is asked to find two free DECtape transports. Transports 4 and 5 were free and are now accessible only to the user typing the ASSIGN command.

The monitor may type back one of the three following error messages:

LOGICAL NAME ALREADY IN USE, DEVICE X ASSIGNED	The logical name specified is the same as a logical given in a previous ASSIGN command. Nevertheless, the device is reserved but without a logical name specified.
ALREADY ASSIGNED TO JOB N	Another user is using the device.
NO SUCH DEVICE	Either the physical device name is misspelled or (in the case of searching for a multiple device) all transports are in use.

Examples

User types:	ASSIGN DTA, ABC	
Monitor responds:	DEVICE DTA6 ASSIGNED	(successful)
User then types:	ASSIGN DTA, DEF	(find another transport)
Monitor responds:	NO SUCH DEVICE	(all in use)
User then types:	ASSIGN PTP, ABC	(reserve the paper tape punch)
Monitor responds:	LOGICAL NAME ALREADY IN USE	(punch is reserved but ABC still refers to DTA6)
	DEVICE PTP ASSIGNED	
User then types:	ASSIGN DTA1, DEF	
Monitor responds:	ALREADY ASSIGNED TO JOB2	(another user has reserved DTA1)

See the DEASSIGN console command for relinquishing devices.

Console Commands for Starting a Job

The following three commands must always be typed in order to start a CUSP or user's program. The initialization commands discussed previously need not necessarily be used; but IJOB is desirable to cancel whatever the previous user at the same console had done.

CORE

Assign CORE memory space. The CORE command requires one argument, a decimal integer, specifying the total number of 1024-word blocks of core memory needed to run his program. The core assigned becomes inaccessible to all other users.

If the core assignment is successful, the monitor makes no comment. If there is not a free block of core of the size requested, the monitor responds with:

N FREE 1K BLOCKS LEFT, NONE ASSIGNED

where N is the number of available 1K blocks of core. When this message occurs, the user is left with no core assigned even though he may have had core assigned when he typed the command.

Sometimes the monitor cannot satisfy a core request even though enough core is available to do so. This problem is best illustrated by an example. Suppose that at the start of the day, two users of a 32K PDP-6 system request 9K of core apiece. With a 6K monitor, only 8K of core remains. Then suppose that the user who was assigned the first 9K block relinquishes his core. The memory available for the user's programs then consists of a free 9K block, an in-use 9K block (by the second user), and a free 8K block. A request for 10K of core cannot now be satisfied because the monitor cannot move programs to join two smaller free blocks. It is up to each computer installation to develop ground rules for the use of core. In this particular instance, the remaining user must save his program, request 9K of core (using a "CORE 9" console command), and restore his program. The monitor assigns the lowest 9K block available, thus making the 17K of unused core into one long block.

To increase his core allotment, the user must retype the CORE command giving the total amount of core required. Similarly, to decrease the amount of core (or to relinquish all core) the CORE command must be retyped. When changing the amount of core assigned, the new core area is always the lowest block of core available that is large enough to fill the request. Thus, the user does not necessarily get back any of the same core with which he started.

If a user interrupts his program (by typing 'control' C) while I/O operations are in progress and types:

CORE 0

all files are closed, end-of-file is written on output magnetic tapes, and all devices not reserved by the ASSIGN console command are returned to the common pool. If instead, the user types:

CORE N

where N is not zero, all devices in current use may be permanently and irretrievably lost to the system. The only recourse is to use one of the error restart sequences described in Appendix 4. A second use of the CORE command which may cause the monitor to fail is typing:

CORE N

while the user program is actually running. (This is possible through the use of STARTC or CONTC described below.)

GET Load the user's core area with a core image that was previously dumped. This console command accepts two arguments: a device name and a file name. The device and file names specify where to find a previously saved program. (A file name extension of "DMP" is assumed.) This command is used both for loading CUSP programs from the system library and for retrieving previously saved user programs.

Upon completion of loading a program (usually after a moderate delay), the monitor types:

JOB SETUP

The job is fully loaded but not started. Other messages indicate error conditions. These are:

NO CORE ASSIGNED	The user has 0 blocks of core
N 1K BLOCKS OF CORE NEEDED	The user has not assigned enough core to load his program. He must now type CORE N and retype the GET command.

FILE NOT FOUND	The file name typed is not found in the directory for the specified device.
DEVICE NOT AVAILABLE	Another user has control of the specified device.
TRANSMISSION ERROR	A parity or other error condition occurred during data transmission.

START — START the user's program running. This command starts execution of the user's program at a location specified in the job data area. The console is taken out of the monitor mode and placed in the user mode so that succeeding input is directed to the user's program. No check is made to see if a program is actually loaded. START accepts an optional argument, an octal number which, if specified, is taken as the starting address instead of the address found in the job data area. The only error condition is:

NO CORE ASSIGNED

Console Commands for Bookkeeping and Debugging

The following commands are most often used during the running of the user's program for debugging or bookkeeping purposes.

SAVE — SAVE the user's program on a dump file for later restoration using the GET command.

If DDT is loaded, the entire user's core area starting at location 75 (JOBCDP) is dumped, including DDT's symbol table. If DDT is not loaded, only the core up to the contents of JOBFF is dumped.

The SAVE console command requires the same two arguments as the GET console command. The user's program, the starting addresses of the user's program and DDT, and other critical parameters are saved in a dump file on the specified device. The file name is the one specified by the typist, and the file name extension is always "DMP." During operation of the SAVE command, all I/O devices are released and the state of the accumulators and program counter is lost. Therefore, after performing a SAVE operation, the user must restart his program from the beginning.

If the SAVE command is successful, the monitor types out (after a moderate delay):

JOB SAVED

Other messages indicate error conditions. These are:

NO CORE ASSIGNED	There is no core.

N 1K BLOCKS OF CORE NEEDED	The user has assigned less core than is to be dumped according to the contents of JOBFF.
DEVICE NOT AVAILABLE	Some other user is using the specified device.
DIRECTORY FULL	No room is left in the directory to enter the new file name.
TRANSMISSION ERROR	Some error condition, such as a bad spot on the tape, occurred during data transmission.

DDT Start the user's debugging program, <u>DDT</u>. The starting address of DDT, the user's debugging program, is a parameter kept in the job data area. When the DDT console command is typed, the monitor starts the user's program at the starting address of DDT. The console is taken out of the monitor mode and placed in the user mode so that input is directed to DDT.

Error messages are:

NO CORE ASSIGNED	The user has no blocks of core.
NO DDT	The starting address of DDT, as stored in the job data area, is 0; therefore DDT is not loaded.

CONT <u>CONT</u>inue execution. When a user's program (including DDT) is stopped by typing 'control' C (C while holding down the 'control' key), execution is resumed by typing the CONT console command. If the user's program stops running because of an error condition or if the user's program deliberately stops by calling "EXIT," the CONT command does not restart the job.

Error messages are:

NO CORE ASSIGNED	
CAN'T CONTINUE	Either the program was never started or it was stopped by other than 'control' C.

<u>Console Commands for Terminating a Job</u>

The following commands are most often used at the end of a computer run when the user is about to leave his console. These commands are also used as a convenient means for relinquishing facilities during a session on the computer.

NOTE: Never abandon a time-sharing console with facilities attached. By so doing, those facilities are lost to other users for the remainder of the day.

DEASSIGN	Return I/O devices to the common pool. This command cancels device reservations made via the ASSIGN command. The DEASSIGN command may be typed alone or with one argument. When an argument is typed, it must be the logical or physical name of some device that was previously reserved by the ASSIGN command. If no arguments are typed, all devices currently reserved by user via ASSIGN commands are affected. The DEASSIGN command may be typed even though the user's program continues to use the devices affected.

Error messages are:

NO SUCH DEVICE	The specified device does not exist.
DEVICE WASN'T ASSIGNED	The specified device was not reserved by the ASSIGN command.

KJOB — Kill the JOB. This console command does the following:

1. Deassigns and stops all I/O devices connected to the job.
2. Returns all the user's core allotment to the monitor.
3. Places the console in the detached state.
4. Makes the job number assigned to the killed job available to the next user who initializes a job.

Console Commands for Starting More Than One Job

The following commands are useful for starting and monitoring more than one job from one console. These commands may also be used for starting a job and relinquishing the time-sharing console to another user.

STARTC — START the user's program as in the START command, but leave the console in the monitor Command mode. The STARTC command differs from the START command in only one respect. STARTC leaves the console in the monitor mode so that further monitor commands may be typed while the user's program runs.

Error messages are the same as for START.

CONTC — CONTinue running the user's program with the console in the monitor Command mode. CONTC bears the same relation to CONT that STARTC bears to START. The CONTC command is often used following a sequence such as: a START command; a control statement typed to the user's program in order to initialize the computation; 'control' C. CONTC allows the computation to continue while the user types additional console commands to start other jobs such as I/O conversions or edits.

Error messages are the same as for CONT.

DETACH Place the console in the DETACHed mode. The DETACH command disconnects the user's console from the job, placing the console in the detached mode but not affecting the status of the original job. In the detached state, the user may initialize another job by typing IJOB, PJOB, CORE, or ASSIGN. In combination with the STARTC or CONTC command, the user can initiate two or more jobs.

There are no error conditions or acknowledgements. However, when the user tries to initialize a new job following a DETACH console command, he may discover that the job capacity of the monitor is exceeded. The introduction to this chapter explains the job exceeded condition. See the explanation of ATTACH below for corrective action.

ATTACH Reconnect the console to a previously initiated but now detached job. The ATTACH command allows a user to reconnect to a job from which he has detached. The ATTACH command requires one argument which must be the job number of the job to which the user desires to connect. There is no acknowledgement if the ATTACH operation is successful. Following the ATTACH, the console is always in the monitor mode. If the job happens to be running, typing "CONT" places the console in the user mode without affecting the operation of the job. A user may type ATTACH on any console, not necessarily the one from which he detached. It is not necessary to type DETACH before typing ATTACH in order to switch the console between two or more previously initialized jobs.

If an error condition occurs, the console is left attached to the job to which it was connected before the ATTACH was typed. The error messages are:

TTYn ALREADY ATTACHED	Either the job number typed is erroneous and by coincidence is in active use at some other console, or another user has attached to the job number specified.
JOB NEVER WAS INITIATED	The job number typed belongs to an inactive job, possibly because a user has attached to the job and typed "KJOB."
ILLEGAL JOB NUMBER	The job number typed is not the number of a legitimate job, including 0 and numbers which are in excess of the job capacity of the monitor.

NOTE: Do not ATTACH to any job without the consent of user who originated that job.

CHAPTER 4

PROGRAMMING FOR USERS' PROGRAMS

The user's program area is the portion of core memory available to the user when his program is running. The first address of this area is defined by the contents of the memory relocation register when the user's program runs, and the last address is defined by the contents of the memory protection register. The user's area is shown in Figure 2-1.

In order to make the following discussion clearer, it is necessary to define some terms and review briefly the operation of the PDP-6 time-sharing hardware. The user's program runs while the computer is in a special mode called user mode. In this mode, the contents of the memory relocation register in the arithmetic processor are added to each memory address before the address is sent to the memory system. The address before this addition takes place is called the relative address; after the addition the address is called the absolute address. In the following discussion, all addresses in the user's area are relative addresses.

To take advantage of the fast accumulators, accumulator addresses (0 through 17) are not relocated. Thus, relative locations 0 through 17 cannot be referenced by the user's program. The Time-Sharing Monitor saves the user's accumulators in this area while the system is servicing a programmed operator request. The contents of the memory protection register are compared with the eight high-order bits of each relative address as the user's program runs. If the relative address exceeds the contents of the memory protection register, the memory violation flag is set and control traps to the Time-Sharing Monitor. Some instructions trap to absolute location 40 when executed in the user mode. These instructions include JRST instructions which attempt to halt and all undefined operation codes. The action on traps to location 40 is the same as the action for programmed operators in executive mode, with the addition that the user mode is turned off when the trap occurs. Input/output interrupts also turn off the user mode and trap to the absolute locations normally used in executive mode. All numbers in the following discussion are octal unless otherwise indicated.

SYSTEM PARAMETERS IN JOB AREA

The first 140 registers of the user job area comprise the job data area which is reserved for storing specific information concerning the job, such as the starting address of the user's program and the I/O device assignments made for this program. Also in this area are registers for saving accumulators during requests for service to the executive system or while some other user's program is running. An understanding of the functions of the specifically reserved locations is required for efficient use of the services of the

Time-Sharing Monitor. The monitor makes its own copy of critical registers in the job data while the job is running and restores these locations to the original values whenever the user's program becomes inactive (due to waiting for I/O or running another user's program). These registers are labeled "protected" in the description below. Some of the protected locations can be modified through the use of the CALL UUO described later in this chapter. The mnemonic assignments given below are made on the library file, JOBDAT, as internal symbol definitions. User's programs must be written with job data area mnemonics declared as externals and loaded with file JOBDAT. A listing of JOBDAT and the definitions of system UUO's (unused operation codes) appears in the appendix.

SYSTEM PUSHDOWN LIST

In addition to specific-purpose locations described below, the Time-Sharing Monitor maintains a pushdown list in the user's area for general purpose temporary storage. All the temporary storage required for serving each user is contained within the user's area and thus, the Time-Sharing Monitor can switch to serving another user at any time without forgetting what it was doing for the original user.

JOBAC	The block of core memory from user's location 0 through 17 cannot be referenced by the user's program because the arithmetic processor substitutes absolute registers 0 through 17. The block of 20 registers starting at JOBAC, defined as user's 0, stores the user's accumulators while servicing Executive requests via the programmed operators.
JOBDAC	A second block of 20 registers stores the accumulators while a job is inactive. Since a job may become inactive either while the user's program is computing or while the Executive program is processing a user's request, the information saved may belong to either the user's program or the Executive program.
JOBUUO, JOB41	Two registers at user's 40 and 41 are used for processing user's programmed operators (001 through 037). The Time-Sharing Monitor reserves programmed operators 040 through 077 for various purposes. The remaining programmed operators (except 000 which is considered illegal) are executed as described in the PDP-6 Programming Handbook using user's locations 40 and 41 instead of absolute 40 and 41. JOBUUO is user's location 40 and JOB41 is location 41.
JOBPC	The location JOBPC saves the program counter and arithmetic processor flags while the job is inactive.

JOBPDP	Protected. The Executive program maintains a pushdown list in the last 16 locations of the user's job data area. JOBPDP contains the pushdown pointer to this pushdown list.
JOBREL	Protected. The left half of JOBREL is zero. The right half contains the highest relative location available to the user, in particular, the contents of the memory protection register.
JOBUXT	Protected. JOBUXT contains the program counter and processor flags for returning to the user's program after the Executive is called by a programmed operator.
JOBLEV	Protected. JOBLEV contains the contents of absolute location 40 when the job is inactive.
JOBSAV	Protected. JOBSAV is a general purpose temporary register for use by the programmed operator processor.
JOBPOV	Protected. JOBPOV contains an address in the user's program to which to jump if a pushdown list overflow trap occurs. If JOBPOV contains 0, no jump occurs, but the monitor stops the user's program and prints an error message on the user's console.
JOBJDA	Protected. A block of 20 registers starting at JOBJDA contains the correlation between input/output devices and the device channel assignments requested by the user's program by means of the INIT prog. op.
JOBDDT	Protected. JOBDDT contains the starting address of DDT, the user's debugging program. When the user types DDT on his console, the monitor finds the starting address of DDT in this location. If JOBDDT contains zero, it is assumed that DDT is not loaded.
JOBSYM	JOBSYM contains a pointer to the symbol table created by the Linking Loader. DDT uses this symbol table for printing and interpreting symbolic addresses. The left half of JOBSYM contains a negative count of the length of the symbol table and the right half contains the lowest register used.

JOBSA	JOBSA contains two addresses. The left half contains the highest register used by the program plus one. The user's input/output buffers are placed in the area immediately following this highest register. The right half contains the starting address of the user's program. Initially, this address is set to the starting address seen by the Loader.
JOBFF	The right half of JOBFF contains the address of the first free location following the user's program. This register is updated by the input/output service routines whenever an input/output buffer is created. The user's program should restore the contents of JOBFF after finishing all use of a buffer in order to recover the core memory used. The left half of JOBFF is not used and normally contains zero.

The remaining locations in the job data area are reserved for future expansion and development. Proposed usage for some of these locations are:

JOBTEM	Two locations beginning with JOBTEM to be reserved for general purpose temporary system storage.
JOBINF	The location JOBINF would contain the job number assigned to the user in bits 0 through 11.
JOBCDP	JOBCDP would contain a byte pointer that points to a command string. The Time-Sharing Executive would accept a command typed on the user's Teletype and determine which system program must be called to process the command. The Executive would then load the system program and set up JOBCDP to point to the command string. The system program then examines the command string, one character at a time, and performs the command operation.
JOBTRP	A block of 20 locations beginning with JOBTRP will contain instructions to be executed when an input/output error or unusual condition occurs. One location is associated with each device channel. If the trap location contains 0, the program is dismissed if an attempt is made to trap on that channel. Otherwise, a JSR instruction is presumed to be in the trap location and control is transferred to the specified trap subroutine as if by a JSR regardless of what instruction is actually at the trap location.

JOBUSY	JOBUSY will contain a pointer to a list of symbols that are defined at the end of the loading process. The user will be able to define all such symbols by means of DDT. DDT will automatically adjust each location where the newly defined symbols are called for.

In addition to the named locations above, all locations between JOBFF and 137 inclusive are reserved for the system pushdown list. The system program loader always starts loading at user's location 140.

USER INPUT/OUTPUT INSTRUCTIONS

The input/output service routines provide the input/output facilities for programs running under the PDP-6 Time-Sharing Monitor System. The object of these routines is to simplify coding for the various input/output devices as much as possible and yet to provide flexibility for performing the most sophisticated functions of input/output. Full advantage is taken of the program interrupt facility of the PDP-6 to overlap in/out operations with computation.

The user's program communicates with the I/O service routines by means of the programmed operators. The terms programmed operator (prog. op.) and unused operation code (UUO) are synonymous. Typical prog. op. functions are to set up a buffer area, to transmit a buffer full of data to an output device, or to release a device for use by other users.

Ring Buffers

Core memory serves as an intermediate buffer between the user's program and the in/out device. An I/O buffer consists of a header block (three words for bookkeeping) and a data storage area. The storage area is subdivided into one or more individual buffers arranged in a ring. To be specific, assume that the device is an input device. Once input operations are under way, the I/O service routines fill a buffer, make the buffer available to the user's program, advance to the next buffer, and begin to fill it if it is free. After filling the last buffer in the storage area, the next buffer to be filled is the first buffer in the area; hence the name ring buffer.

The user's program follows along behind, emptying the next buffer if it is full, or waiting for the next buffer to fill up if it is not yet full.

For output, the user's program and the I/O routines exchange roles, the user filling the buffers and the I/O routines emptying them.

Buffer Structure

The buffer structure consists of two separate segments, the header and the storage area. The header consists of the following three words.

Word 1	The right half contains the address of the second word of the buffer currently in use by the user's program. Bit 0=1 means no input or output has occurred for this ring.
Word 2	A byte pointer to the last item referenced in the buffer.
Word 3	A count of the number of bytes remaining in the buffer.

For most applications, the user's program finds all the information it requires about the state of the buffer area in the header. When referring to a buffer by name, the address of the buffer header, not the storage area, is implied.

The storage area is subdivided into a number of individual, equal-length buffers. The size of each buffer depends on the requirements of the device serviced by the buffer. The first two words of each buffer are reserved for special functions; the first two words are:

Word 1	Unused, but reserved for a block number from fixed address devices such as DECtape or drum.
Word 2	In the right half, the address of the second word of the next buffer in the ring. In the left half, bit 0 is a flag as explained below; bits 1 through 17 contain the buffer size in words (excluding these two words of overhead).

Bit 0 of the second word of the buffer, called the use bit, is a flag that indicates whether the buffer contains active data. This bit is set to 1 by the I/O routines when the buffer is full on input (or being emptied on output) and set to 0 when the buffer is empty on output (or being filled on input). The use bit prevents the I/O routines and the user's program from interfering with each other by attempting to use the same buffer simultaneously.

In all data processing modes, the first data word of the buffer is reserved for a count of the number of data words in the buffer (excluding itself). Thus, a buffer may be only partially filled. The left half of this word is reserved for other bookkeeping purposes depending on the particular device.

Job Initialization

To clear all previous device assignments and prepare the I/O service routines for receiving further initialization commands, execute the instruction:

CALL [SIXBIT /RESET/]

This must be the first instruction in each program. RESET instantly stops any I/O operations that the job has started and, thus, is the only method for releasing a device without waiting for I/O completion. This function is necessary when, for example, the line printer is out of paper and cannot finish printing. RESET also reclaims the buffer area by setting the contents of JOBFF to the address contained in the left half of JOBSA.

Device Initialization

INIT Prog. Op.

A device is initialized and buffer headers specified by executing the command:

```
INIT D, MODE
SIXBIT /device name/
XWD OBUF,IBUF
error return
normal return
```

INIT performs the following functions:

1. The 4-bit channel number D is assigned to the device name appearing in the SIXBIT statement. Henceforth, the I/O service routines interpret the number D as referring to that device. The device name is either a logical or physical name with logical names taking precedence (see "ASSIGN" console command).

2. The data processing mode is selected by mode (see below).

3. The buffer area headers are specified to the device service routines. Each header must be three locations reserved by the user in his program. Only those headers which are to be used need be specified; the output header need not be specified if the program performs only input from a device.

4. The buffer header block is initialized as follows: The first and third words are set to zero. The left half of the second word is set up with an appropriate byte-pointer size field.

INIT Data Modes

One item of information specified in the INIT command is the data processing mode. The following modes are available:

Octal Code	Mnemonic	Meaning
0	A	ASCII, 7-bit characters, packed left justified by the PDP-6 byte instructions.
1	AL	ASCII line, same as A except that the buffer is terminated by a form-feed, vertical tab, line-feed or ALT mode.
10	I	Image, a device dependent mode, the buffer is filled with unprocessed data exactly as supplied by the device.
13	IB	Image binary, 36-bit words. This mode is similar to binary mode except that no automatic formatting or checksumming is done by the service routines.
14	B	Binary, a blocked format consisting of a word count (the first word of the buffer) followed by N 36-bit data words. Checksumming is done automatically by the service routines or by the device itself. The word-count N is in the right half of first word of each block, and if a checksum is computed (cards, paper tape), it is in bits 6-17. This 12-bit folded checksum is computed by summing the 36-bit data words in the block (2's complement add), followed by a 1's complement sum of bits 0-11, 12-23, 24-35.
16	DR	Dump respecting record boundaries. Data is transmitted between any contiguous block of core and one record on the I/O device. In dump mode, ring buffers are not used. The control for the size of the data block and the location in core memory is from a command list explained below.
17	D	Dump, same as DR except that record boundaries are ignored on input and arranging the data into standard-length records is automatic on output.

If the INIT prog. op. specifies a mode that is not appropriate for the given device, the user's job is stopped and the following error message is typed on the user's console:

```
ILL DEVICE DATA MODE FOR DEVICE X AT USER LOC N
```

where X is the physical name of the device and loc N is the address of the INIT (or SETSTS described below) prog. op.

INIT Control Modes

In addition to the four bits to specify one of the data modes described above, some of the remaining address bits of the INIT prog. op. specify special control functions. Because most of these bits specify device-dependent processing and do not perform the same function for all devices, only the bits that are defined for all devices are described below. The other bits are explained in the separate discussions for each device.

Bit	Name	Action
31	IOWC	I/O word count. If a 1, forces the service routines to accept the word count in the first data word of the buffer (as computed by the user) instead of computing a word count from the contents of the byte pointer in the buffer header (output only).
30	IOCON	I/O discontinuous mode. If a 1, specifies stopping the device after each buffer is filled (input only).

A device may be initialized separately on two device channels if one INIT prog. op. specifies the output buffer header and the other specifies the input buffer header. All further prog. ops. may then refer to either channel number; the OUTPUT prog. op. can use the channel that was initialized for input, for example.

If any of the I/O prog. ops. that require a device channel specification are given (except INIT, CLOSE, and RELEAS as explained below) before a device is actually initialized on that channel (by the INIT prog. op.), the monitor stops the user's program and types the following error message on the user's console:

I/O TO UNASSIGNED CHANNEL AT USER LOC N

where N is the address in the user's area of the offending prog. op.

Buffer Initialization

INBUF Prog. Op.

To set up a ring buffer for input data, a program executes the prog. op.:

INBUF D,N

where N specifies an N-buffer ring and D specifies a device channel number (the INIT prog. op. must be used previously).

The storage space for the buffer ring is taken from the free storage following the program. The location JOBFF (a permanent global symbol defined on the system symbol file, JOBDAT) contains the address of the first free location following the program. The contents of JOBFF are updated after each INBUF prog. op. If the user wishes to abandon a buffer for one device and set up a buffer for another device, the program should restore the contents of JOBFF to the original value. Then the I/O service routines can reuse the old buffer area in core memory for setting up the new buffer. The RESET operation explained above also resets JOBFF to its initial value calculated when the program is loaded.

OUTBUF Prog. Op.

An ouput buffer area is set up by:

```
OUTBUF D,N
```

If no buffer area is set up when the first input or output command is issued, a two-buffer ring is set up automatically from the free storage following the program.

File Initialization

LOOKUP and ENTER Prog. Ops.

These programmed operators initialize a file of data. An ENTER must be given before output, and LOOKUP before input. These prog. ops. have no effect for single-file devices such as paper tape or Teletype input/output. With a multiple-file device, DECtape in particular, these programmed operators specify a file name extension for the file. The format for LOOKUP and ENTER is:

```
LOOKUP D,E          ENTER D,E
error return        error return
OK return           OK return
```

D is the device number assigned to the DECtape. E is the first address of a 4-word directory entry block that must be reserved by the user. For both LOOKUP and ENTER, the user's program must put the desired file name in sixbit code into E, the first word of the directory block. Likewise, the file name extension must be put into the left half of the word at E+1. For ENTER only, the date may be placed in the rightmost 12 bits of E+2.

The monitor action on ENTER is:

1. The monitor searches the directory of the specified device for a matching file name and file name extension. If no extension is specified (the extension bits are zero), the file name extensions are excluded from the directory search.

2. If a matching entry is found, the new entry replaces the old. Otherwise, the new entry is appended to the directory.

3. File retrieval information is added to the directory entry. The date, if not already specified, is inserted in the third word of the directory entry.

4. The I/O service routines are initialized to begin outputting at the location specified by the directory entry.

5. If the directory is full and there is no room for appending the new entry, the monitor takes the error return. If the entry is successful, the monitor takes the OK return.

The monitor action on LOOKUP is:

1. The monitor searches the directory as for ENTER.

2. If no matching entry is found, the monitor takes the error return.

3. The complete four-word directory entry is copied from the device directory into the four-word block at E.

4. The file retrieval information is given to the I/O service so that inputting may begin at the beginning of the file.

5. If the LOOKUP is successful, the monitor takes the OK return.

Data Transmission

The two prog. ops., INPUT and OUTPUT, are used to synchronize I/O operations with the user's program. The format for INPUT and OUTPUT is:

```
INPUT D,0                OUTPUT D,ADR
```

D is the device number assigned to the desired device. ADR, optional and usually zero, is explained below.

The functions of these two prog. ops. are similar, one serving for input operations and the other for output operations. The commands operate as follows:

INPUT Prog. Op. (Non-Dump Mode)

This prog. op. releases the current buffer to the I/O routines for refilling and advances the buffer header pointers to the next buffer. If the next buffer is not full when the input prop. op. is given, the I/O routines delay until the buffer is full or an end-of-file indication is received. Since the input device normally continues until all buffers are full, the input prog. op. serves to synchronize the user and the input device.

OUTPUT Prog. Op. (Non-Dump Mode)

This prog. op. computes a word count from the position of the byte pointer in the buffer header, stores the word count in the right half of the first data word in the buffer, and makes the buffer available to the device for outputting. The buffer header pointers are advanced to the next buffer, and the item count is set to the maximum number of bytes that can fit in the buffer. The service routines assure that the new buffer is empty by waiting until it is output, if necessary, and then clear the buffer to zero.

If ADR is nonzero, it is taken as a pointer to the word of the next buffer. The sequence of buffers in the ring thus can be altered. The buffer pointed to by ADR can be in an entirely separate ring from the previous buffer. Once a new buffer position is established, the following buffers are taken from the ring starting at ADR. When using this feature in output operations, the I/O routines compute an improper word count unless one of the following precautions is taken:

1. In the INIT prog. op. have bit 31 a 1 to prevent the service routines from computing a word count (useful if the user's program computes its own word count and places it in the right half of the first data word of the buffer or if outputting directly from an input buffer).

2. Set the byte pointer of the buffer header to point to the last item in the new buffer.

Examples

When the INIT command is given, the item count in each buffer header is set to zero and the byte size is set according to the data processing mode. The INBUF and OUTBUF commands set the first word of the buffer header to point to the first buffer in the ring. The remaining header words are set up when the first input or output command is given. If the item count is ignored by the user's program during output operations, the output prog. op. must be given once following the INIT prog. op. to set up the output buffer header. This one appearance of OUTPUT does not cause data transmission to the device. If the user's output subroutine checks the item count before putting data into the output buffer, this initial output command happens automatically.

A typical Get One Character subroutine from an input device is:

```
GET:        0
            SOSG IBUF+2          ;DECREMENT ITEM COUNT AND TEST
            INPUT D,             ;IF NO DATA, CALL INPUT
            ILDB AC,IBUF+1       ;GET NEXT CHARACTER IN AC
            JRST @GET            ;EXIT
```

A typical Output One Character Subroutine would be:

```
PUT:        0
            SOSG OBUF+2          ;DECREMENT ITEM COUNT AND TEST
            OUTPUT D,            ;IF NO ROOM LEFT CALL OUTPUT
            IDPB AC,OBUF+1       ;DEPOSIT CHARACTER FROM AC
            JRST @PUT
```

INPUT and OUTPUT Prog. Ops. (Dump Mode)

When using dump mode (either 16 or 17, D or DR), the ring buffer scheme is entirely eliminated. Instead, input and output are done directly from the user area. Control of I/O operations is via a command list. The location of the command list is given by the INPUT or OUTPUT prog. op. in the format:

```
INPUT D,ADR          OUTPUT D,ADR
```

D is the device number assigned to the desired device.

Command List Format

The command list format varies for the specific device but is in general:

```
IOWD A,B          ;TRANSFER A WORDS STARTING AT CORE LOCATION B.
XWD 0, A          ;GO TO A FOR THE NEXT COMMAND
0                 ;TERMINATE THE COMMAND LIST
```

The command list is assumed to be in sequential locations except when a GOTO word is encountered. The input/output service routines delay until the command list is completely processed before returning control to the user's program.

Status Checking and Setting Prog. Ops.

Certain errors, such as parity or checksum errors, can occur during data processing. For each 4-bit device number, a status word is provided which contains error indication bits and other useful conditions. The status word is examined or tested by the commands:

```
GETSTS D,ADR or STATZ D, MASK or STATO D,MASK
```

These three prog. ops. are exactly analogous to the PDP-6 instructions CONI, CONSZ, and CONSO except that D is the 4-bit device channel number.

Status Bits

The bits of the status word have the following meanings:

Bit	Meaning
18	IOIMPM, Improper Mode. The selected mode is unobtainable.
19	IODERR, Device Error. The device's self-checking circuits indicate an error (parity failure,etc).
20	IODTER, Data Error. The computed checksum failed or invalid data was received.

Bit	Meaning
21	IOBKTL, Block Too Large. A block of data from a high-speed device (DECtape, mag tape, drum) was too large to fit in one buffer.
22	IODEND, Data End Encountered. End-of-file.
23	IOACT, Device Active and Currently Transmitting Data.

Bits 26 through 35 read back the mode transmitted to the I/O service routines by the most recent INIT or SETSTS (see below) prog. op. Bits 24 and 25 give device-dependent conditions that are explained below for each particular device.

At times it is necessary to change certain bits in the status word; for example, clearing error bits or the end-of-file bit, or changing data modes during I/O operations. The prog. op.:

```
SETSTS D,BITS
```

delays until the device on device channel D stops transmitting data, then replaces the status word with BITS. All bits of the status word are affected by this prog. op. except bit 23 (IOACT). If the data mode is changed by this prog. op., the byte pointers in the buffer headers are changed appropriately. Note that there is only one status word per device even though a device is initialized on more than one device channel.

Terminating a File

To terminate a file use the CLOSE prog. op. The format is:

```
CLOSE D,
```

CLOSE finishes output operations on device channel D by dumping the last buffer if it is partially full and by writing an end character if the device requires one. For input operations, all buffers are emptied by setting the use bits to zero and the item count in the buffer header to zero. For either input or output, the device is returned to the same state as before the first input or output command.

CLOSE does not necessarily terminate the input file and the output file if the device is 2-way. If the OUTPUT prog. op. was used on channel D, the output file is terminated when CLOSE is used on channel D. Similarly, the input file is terminated only if the INPUT prog. op. was used. Also, the user's program may deliberately inhibit the termination of either the input or output file by using CLOSE with the following format:

```
CLOSE D,N
```

If N is equal to 1, the output file is not terminated regardless of whether an OUTPUT prog. op. was used. If N is equal to 2, the input file is not terminated. If N is equal to 3, neither file is terminated and the only action is a delay until the device becomes inactive.

Relinquishing a Device

When all I/O to a device is finished, the user's program must give the prog. op.:

```
RELEASE D,
```

RELEASE performs a CLOSE operation unless CLOSE was used previously. Then the device on channel D is made available for other users except when the device is reserved by the ASSIGN console command. A new INIT prog. op. must be given if further operations with the device are required.

If a device is initialized on one channel for input and another for output, only the appropriate channel is affected by RELEASE.

I/O Synchronization

In some instances it is desirable to delay until a device has completed its input/output activities. For example, a program that backspaces magnetic tape might follow up each backspace with a check for the load-point status bit. Because the backspace prog. op. returns to the user before the operation is complete, the user's program must somehow delay in order to check for the load-point status at the end of the backspace operation rather than during the middle. The following prog. op. provides the delay function:

```
CALL D, [SIXBIT /WAIT/]
```

This prog. op. does not return control to the user's program until the device on channel D becomes inactive.

Device Characteristics Checking

The following prog. op. allows a user's program to determine a great deal of information about a device both before and after initializing the device. The sequence:

```
MOVE AC, [SIXBIT /NAME/]
CALL AC, [SIXBIT /DEVCHR/]
```

loads into accumulator AC a word describing the characteristics of the device having the physical or logical name NAME. Bits in the device characteristic word are interpreted as follows:

Bit	Meaning
22 through 35	Data modes are permitted for this device. Mode J is permitted if bit 35-J is a 1.
21	If 1, this device is reserved for some job by an INIT prog. op.
20	If 1, this device is reserved for some job by the ASSIGN console command.
17	If 1, this is an output device.
16	If 1, this is an input device.
15	If 1, this device has a directory (DECtape is currently the only such device). If 0, the prog. ops. ENTER and LOOKUP act as SKIPA instructions.
14	If 1, this device is a Teletype.
13	If 1, this device is magnetic tape. If 0, the MTAPE prog. op. acts as a no-op.
12	If 1, the device is accessible to this job by an INIT prog. op.. However, the device may become unavailable between the DEVCHR and the INIT prog. op. unless bit 21 or 20 is already 1.
6 through 11	Image mode byte size; 0 if image mode is not permitted.
5	If 1, the device is a Teletype that is in use (someone has typed on it). If 0, the device either is not a Teletype or is in the detached mode.
4	If 1, this device is the user's console Teletype. If 0, this device is not the user's console Teletype. This bit is set to 0 when the user types the DETACH command.
3	If 1, this device is the line printer which accepts special carriage control characters as described in the PDP-6 Handbook, F-65.
2	If 1, this device is a card reader for which each line is filled out to 80 characters with spaces.
1	Unused.
0	If 1, the device is a DECtape for which the directory is in core.

Bits 0, 4, and 5 are intended for the use of the monitor rather than for providing useful indications for the user.

If the user's program requests the characteristic word for a nonexistent device name, the word returned is zero.

DEVICE DEPENDENT FUNCTIONS

The paragraphs below explain the unique features of each I/O device. All devices accept the prog. ops. explained above unless otherwise specified. Buffer sizes below are octal and include the two bookkeeping words. Table 4-1 is a summary of the characteristics of all devices.

TABLE 4-1 DEVICE SUMMARY

Physical Name	Name	Hardware Type Number	Prog. Op.	Data Modes	Buffer Size (octal)
CTY	Console Teletype	626	INPUT OUTPUT	A, AL	23
TTY1, TTY2, ... , TTY77	Teletype	630	INPUT OUTPUT	A, AL	23
PTR	Paper Tape Reader	760	INPUT	A, AL, IB, B, I	43
PTP	Paper Tape Punch	761	OUTPUT	A, AL, IB, B, I	43
LPT	Line Printer	646	OUTPUT	A, AL	34
CDR	Card Reader	461	INPUT	A, AL, B, I	36
DTA1, DTA2, ... , DTA0,	DECtape	551/555	INPUT OUTPUT LOOKUP ENTER USETO USETI UGETF CALL [SIXBIT /UTPCLR/]	A, AL, IB, B, I, DR, D	202
MTA0, MTA1, ... , MTA7	Magnetic Tape	516	INPUT OUTPUT MTAPE	A, AL, IB, B, I, DR, D	203

Teletype

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

Line number N of the Type 630 Data Communications 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. An idle line is considered nonexistent. For interconsole communication, 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" or "IJOB" is the only way to reassign a Teletype that has been deassigned.

Buffer Size - 23

Data Modes A. ASCII - All characters typed in appear in the input buffer as typed with the following exceptions:

RUBOUT	erases the previous character. Successive rubouts erase characters to the left until the beginning of the current bufferful. For each character erased, a backslash is typed. If there is no character to erase, a carriage-return/line-feed is typed.
RETURN	is followed automatically with line-feed, both of which appear in the input buffer.
'control' U	types back as ↑U followed by carriage-return/line-feed. This character deletes the entire current bufferful of ASCII characters.
'control' O	similar to 'control' U but has special action during output.
'control' P	does not appear in the input buffer. This character informs the I/O service routine that there is a Type 35 Teletype on the line. The difference between the two Teletype models is that the Type 35 has a tab, form-feed, and vertical tab mechanism that the Type 33 does not have. Alternate uses of 'control' P turn the special "Type 35" mode on and off.
'control' Z	types as ↑Z and appears in the buffer as 032. This character serves as an end-of-file.

TAB	echoes as an appropriate number of spaces to place tab "stops" at every eighth space unless 'control' P is typed.
VT	(vertical tab) types as four line-feeds.
FORM	(form-feed) types as eight line-feeds.
'control' C	types as ↑C followed by carriage-return/line-feed. This character places the console in the monitor mode, ready to accept monitor commands.

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

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 'control' C, the console is placed in monitor mode. If it is 'control' O, all Teletype output buffers that currently are 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 'control' C.

AL. ASCII Line - All input character processing is the same as for the ASCII mode. However, the buffer is terminated on RETURN (carriage-return), LINE FEED, FORM, VT, "control Z", and ALT (alt mode, (175) sometimes labeled ESC, escape). One of these characters always appears at the end of each bufferful. RETURN is always followed by LINE FEED so that RETURN is never the last character in the buffer.

On output, AL mode is no different from the A mode.

DDT Submode

In order to allow a user's program and the DDT debugging program to both use the console Teletype without interfering with one another, the Teletype service routine provides the DDT submode. This mode is entered by the prog. op.:

```
CALL [SIXBIT/DDTGT/]
```

and left by the prog. op.:

```
CALL [SIXBIT/DDTRL/]
```

and does not affect the Teletype status if it is initialized with the INIT prog. op. It is not necessary to use the INIT prog. op. in order to do I/O in the DDT submode. I/O in DDT mode is always to the user's console 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 a bufferful 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. This prog. op. 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 'control' O and 'control' 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 to output the string, and restarts the user's program. Character processing is the same as for ASCII mode output.

Paper Tape Reader

Device Name - PTR

Buffer Size - 43

Data Modes (Input Only)

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. Turning the reader off serves as an end-of-file and results in the character 032 ('control' Z) appearing in the buffer.

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

I. Image - There is no character processing. The buffer is packed with 8-bit characters exactly as read from the input tape. Turning the reader off is the end-of-file indication but does not cause a character to appear in the buffer.

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.

B. Binary - Checksummed binary data is read in the following format. The right half of the first word of each physical block contains the number of data words that follow and the left contains half a folded checksum. The checksum is formed by adding the data words using 2's complement arithmetic, then splitting the sum into three 12-bit bytes and adding these using 1's complement arithmetic to form a 12-bit checksum. The data error status flag (IODERR) 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, turning the reader off is the end-of-file indication but does not result in putting a character in the buffer.

Paper Tape Punch

Device Name - PTP

Buffer Size - 43

Data Modes

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.

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

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

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.

B. Binary - Each bufferful of data is punched as one checksummed binary block as described for the paper tape reader.

Special Prog. Op. Service

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

Line Printer

Device Name - LPT

Buffer Size - 34

Data Modes

A. ASCII - ASCII characters are transmitted to the line printer exactly as they appear in the buffer. See the PDP-6 Handbook, F-65, for a list of the vertical spacing characters.

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 including placing a RETURN, LINE-FEED at the end of each line.

Special Prog. Op. Service

The first output prog. op. of a file and the CLOSE prog. op. at the end of a file cause an extra form-feed to be printed to keep files separated.

Card Reader

Device Name - CDR

Buffer Size - 36

Data Modes

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, and an end-of-file character 032 appears as the last character in the input buffer. 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. The left arrow character 137 appears in each column containing an invalid punch. The appendix contains the translation table for punching the special ASCII characters on cards.

AL. ASCII Line - Exactly the same as the A mode.

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 one. 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 with the character 0032 appearing in the buffer as the last character of the file. Cards are not split between two buffers.

B. Binary - Card column one must contain a 7-9 punch to verify that the card is in binary format. The absence of the 7-9 punch results in raising the IOIMPM (improper mode) flag in the card reader status word. Card column two 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 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.

DECtape

Device Name - DTA0, DTA1, ..., DTA7

Buffer Size - 202

Data Modes

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.

AL. ASCII Line - Same as A.

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

IB. Image Binary - Same as I.

B. Binary - Same as I.

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

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 earlier in this chapter under Input/Output Prog. Ops. (Dump Mode), except that the GOTO command is not implemented; any positive number appearing in a command list terminates the list. Dump data is blocked into standard-length DECtape blocks by the DECtape control automatically. Unless the number of data words is an exact multiple of the standard length of a DECtape block (200), after each output prog. op., the remainder of the last block written is wasted. The input prog. op. must specify the same number of words that the corresponding output prog. op. specified in order to skip over the wasted fractions of blocks.

DECtape Format

A standard reel of DECtape consists of 1102 (octal) prerecorded blocks each capable of storing 200 36-bit words of data. Between blocks are recorded block numbers which label the blocks for addressing purposes. These block numbers run from 0 to 1101. Block 0 is normally not used during time-sharing and is reserved for a bootstrap loader. Block 1 is the directory which contains the names of all files on the tape and the block number where each file begins. Blocks 2 through 1101 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 or smaller than 0, the monitor stops the user's program and types the following error message on the user's console:

```
ILL DT BLOCK NO., BUFFER AT X DEVICE DTAY
```

where X is the address of the ring buffer involved and DTAY is the offending DECtape unit.

The directory format is:

Word 0	Left half	Last block number in use.
	Right half	The word number in the directory block where the first directory entry is found (normally 5). Thus, words 1 through 4 are normally unused.

Each directory entry contains:

First word		The file name in sixbit code.
Second word	Left half	The file name extension in sixbit code.
	Right half	The first block number of the file.
Third word	Bits 0-23	Unused (zero).
	Bits 24-35	The date compressed according to the formula: ((year-1964)*12+(month-1))*31+day-1
Fourth word		Unused in nondump files. For dump files requiring a single-word command list, the command list word may be stored here by the user's program.

Unused words in the directory contain zero. Thus, zero marks the end of the list of directory entries. Thirty entries (decimal) fit in one directory.

When a directory search is begun for either LOOKUP or ENTER, the service routine examines the right half of word 0 of the directory. If this half word is not in the range from 1 through 174, the monitor stops the user's program and types the following error message on the user's console:

```
BAD DIRECTORY FOR DEVICE DATAX;EXEC CALLED FROM USER LOC N
```

The file format is:

A file consists of any number of DECtape blocks. A dump file consists of the required number of adjacent blocks, but a nondump file may consist of nonadjacent blocks. For nondump files only, 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	A count of the number of words in this block that are used (maximum 177).
Words 1 through 177		Data packed exactly as the user placed it in his buffer.

Special Prog. Op. Service

Several prog. ops. are provided for manipulating DECtape. These prog. ops. allow the user to manipulate block numbers and to operate upon directories.

Prog. Op.	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	Automatically increments the free block pointer, places the number of the first unused block in user's location E, and does a USETO D,E.
ENTER D,E error return	User's locations E, E+1, E+2, and E+3 must be reserved for a directory entry. The DECtape service routine searches the directory for a file name and extension that match the contents of E and the left half of E+1. If the extension (left half of E+1) is zero, it is not included in the search. If no match is found and four free words are found at the end of 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 nonzero), and copies the resulting four words into the four free words at the end of directory. Then the free block pointer is updated and a USETO is performed. 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 the left half of E+1 (if nonzero) are matched against the corresponding words of each directory entry. If a match is found, the entire 4-word entry is read from the directory into the 4-word block at E. Then a USETI is automatically performed using the first block number of the file which is found in the right half of E+1. If no match is found, LOOKUP returns to the user's program at the next location. Otherwise, LOOKUP skips one location.
CALL D, [SIXBIT /UTPCLR/]	UTPCLR clears the directory of the DECtape on device channel D. A cleared directory has word 0 containing 1 in the left half and 5 in the right half and all other words containing zero. Only the directory block (block 1) is affected by UTPCLR; the other blocks are unaffected. This prog. op. does nothing if the device on channel D is not DECtape.

In addition to the prog. ops. 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 prog. op. 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 word contains zero, the next free block number is placed there, the free block pointer is updated, and the service routine gets set to write in that free block upon the next OUTPUT prog. op. If this half word 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 1, it is not changed and the service routine uses it as the block number for the next OUTPUT prog. op..

For both INPUT and OUTPUT, block 1 (the directory) is treated as an exception case. The "link" in block 1 is actually the free block pointer and, therefore, is not altered on output. Thus, if the user's program gives:

```
USETI D,1
```

to read block 1, it is treated as a 1-block file.

The prog. op. CLOSE places a 1 in the left half of the first word in the last output buffer, thus, terminating the file.

The prog. op. RELEAS writes onto block 1 the copy of the directory which is normally kept in core, 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.

Special Status Bits

If an attempt to write is made on a unit with the write-lock switch on, the device error flag (IODERR) is raised.

Important Considerations

If an attempt is made to use a unit that is turned off or that has no tape mounted, all DECtape units will be hung up until a tape is actually mounted. Good practice is to mount scratch tapes on all transports not otherwise in use.

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 in core is deleted following the next RELEAS unless the user assigns a tape unit to his job using the ASSIGN console command. If the unit is assigned, the DECtape service routine reads the directory from block 1 only once, the first time it is required and thereafter uses the directory image in core memory regardless of the appearance of RELEAS prog. ops. To inform the DECtape service routine that a new tape has been mounted on an assigned unit, the user must use one of the console commands IJOB, KJOB, ASSIGN, or DEASSIGN, whichever is appropriate to the situation. 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.

Magnetic Tape

Magnetic tape format is IBM compatible and is not described here.

Device Name - MTA0, MTA1, ..., MTA7

Buffer Size - 203

Data Modes

A. ASCII - Data is 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 words per record is standard. The word-count is not written on the tape.

AL. ASCII Line - Same as A.

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

IB. Image Binary - Same as I.

B. Binary - Same as I.

DR. Dump Records - 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 earlier in this chapter under Input/Output Prog. Ops. (Dump Mode). 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.

D. Dump - This mode is accepted but actually functions as DR mode 16.

Special Prog. Op. Service

The CLOSE prog. op. 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 prog. op. called MTAPE provides for such tape manipulation functions as rewind, backspace record, backspace file, 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:

Function	Action
1	Rewind to load point
11	Rewind and unload
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

MTAPE waits for the magnetic tape unit to complete whatever action is in progress before performing the indicated function. 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. The MTAPE prog. op. affects only the physical position of the tape and does not change the data that has already been read into the buffers.

Special Status Bits

Special bits of the status word are reserved for selecting the density and parity mode of the magnetic tape. The following bits are set and cleared by the INIT or SETSTS prog. op..

Bit	Name	Action
29	IORCK	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.
27,28	IODENS	I/O density. 00 or 10 =556 bpi 01 =200 bpi 11 =800 bpi
26	IOPAR	I/O parity. 0 for odd parity, 1 for even parity.

The following bits indicate special magnetic tape conditions and are set by the magnetic tape service routine when the conditions occur.

Bit	Name	Meaning
19	IODERR	I/O Device ERRor. When set to one during an output operation means that the write protect ring is out.
24	IOBOT	I/O Beginning of Tape. The tape is at the load point.
25	IOTEND	I/O Tape END. The tape is at or past the end point.

MISCELLANEOUS MONITOR FUNCTIONS

The following monitor functions provide various user services. All of these functions are obtained by the CALL prog. op. using the following format:

```
CALL AC, [SIXBIT /FUNCT/]
```

AC specifies the accumulator that either contains an argument for the function or receives the function value. FUNCT is the name of the monitor routine which performs the action.

Date

The prog. op.:

```
CALL AC, [SIXBIT /DATE/]
```

loads the date into accumulator AC. The date is compressed into 12 bits by the formula:

date=((year-1964)*12+(month-1))*31+day-1

Exit

The prog. op.:

```
CALL [SIXBIT /EXIT/]
```

closes and releases all I/O devices, stops execution, prints the message "EXIT" on the user's console, and places the console into monitor mode.

Setddt

The sequence:

```
MOVEI AC,ADR
CALL AC, [SIXBIT /SETDDT/]
```

places ADR into location JOBDDT of the job data area as the starting address used when the user types the DDT console command. The user's program cannot alter the contents of JOBDDT except by using the SETDDT function because that location is protected by the monitor.

Setpov

SETPOV allows the user to program an error correction procedure for pushdown list overflows. The sequence:

```
MOVEI AC,ADR
CALL AC, [SIXBIT /SETPOV/]
```

places ADR into location JOBPOV of the job data area. On a pushdown overflow, the monitor traps the user's program to location ADR if ADR is nonzero. Location JOBPOV is protected by the monitor and cannot be changed except by the use of SETPOV. If SETPOV has not been used and a pushdown list overflow occurs, the monitor stops the user's program and types the following message on the user's console:

```
PDL OV AT USER LOC N
```

Switch

The prog. op.:

```
CALL AC, [SIXBIT /SWITCH/]
```

places the contents of the console data switches into accumulator AC.

Timer

The prog. op.:

```
CALL AC, [SIXBIT /TIMER/]
```

places the time of day, in 60ths of a second from midnight, into accumulator AC (50ths of a second if the power line frequency is 50 cps.).

Special Loader Function

The following is a very special purpose loader function and is subject to change or deletion with no advance notice. This function is provided for the convenience of the loader. The prog. op.:

```
CALL [SIXBIT /LDRBLT/]
```

does the following:

1. All devices are released.

2. The contents of accumulator 2 (henceforth referred to as the loader offset) are added to the contents of JOBSYM in the offset job data area.

3. The user's program is assumed to be loaded above the location in which it will run by an amount equal to the loader offset. The user's program is block transferred down appropriately. The lowest location transferred is 30, the highest is determined by the contents of the left half of JOBSA in the offset job data area.

4. A SETDDT operation is performed on the contents of JOBDDT in the new job data area.

5. The job is stopped and the message:

   ```
   LOADER FINISHED
   ```

   is typed on the user's console. The console is returned to monitor mode.

6. The contents of accumulator 2 are set to zero.

EXAMPLES

Typical Program Initialization

```
BEG:        CALL [SIXBIT /RESET/]
                .
                .
                .
```

General Device Initialization

```
INIDEV:     0                           ;JSR HERE
            INIT 3,14                   ;BINARY MODE,CHANNEL 3
            SIXBIT /DTA5/               ;DEVICE DECTAPE UNIT 5
            XWD OBUF,IBUF               ;BOTH INPUT AND OUTPUT
            JRST NOTAVL                 ;WHERE TO GO IF DTA5 IS BUSY

;FROM HERE DOWN IS OPTIONAL DEPENDING ON THE DEVICE AND PROGRAM
;REQUIREMENTS

            MOVE 0, JOBFF
            MOVEM 0,SVJBFF              ;SAVE THE FIRST ADDRESS OF
                                        ;THE BUFFER RING IN CASE THE
                                        ;SPACE MUST BE RECLAIMED.
            CALL AC, [SIXBIT /DATE/]
            MOVEM AC,OUTNAM+2           ;PUT THE DATE INTO THE DIRECTORY
                                        ;ENTRY
            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, OUTNAM             ;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
INBUF:      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
```

General Input One Character Subroutine

```
GETCHR:   0                       ;JSR HERE
          SOSLE IBUF+2            ;DECREMENT THE ITEM COUNT
          JRST GETOK              ;DATA IS ALREADY IN THE BUFFER
          INPUT 3,                ;REFILL THE BUFFER
          STATZ 3,740000          ;CHECK THE 4 ERROR BITS
          JRST INERR              ;WHERE TO GO ON AN ERROR
          STATZ 3,20000           ;CHECK END OF FILE BIT
          JRST ENDFIL             ;WHERE TO GO ON END OF FILE
GETOK:    ILDB AC,                ;FETCH CHARACTER FROM THE BUFFER
          JRST@ GETCHR            ;RETURN
```

General Output One Character Subroutine

```
PUTCHR:   0                       ;SIMILAR TO GETCHR BUT NO END
                                  ;OF FILE CHECKING IS REQUIRED
          SOSLE OBUF+2
          JRST PUTOK
          OUTPUT 3,
          STATZ 3,740000
          JRST OUTERR
PUTOK:    IDPB AC, OBUF+1
          JRST@ PUTCHR
```

Simplified I/O Subroutines

```
GETCHR:   0
          SOSG IBUF+2
          INPUT 3,
          ILDB AC,IBUF+1
          JRST@ GETCHR
PUTCHR:   0
          SOSG OBUF+2
          OUTPUT 3,
          ILDB AC, OBUF+1
          JRST@ GETCHR
```

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
```

Halting A Program

```
ENDPRG:     CALL [SIXBIT /EXIT/]              ;CLOSES AND RELEASES ALL DEVICES
```

User-Generated Buffers

The following code illustrates avoidance of the INBUF prog. op. Analogous code may replace OUTBUF. This code also illustrates the operation of 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 NONSTANDARD 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+1,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+1,BUF3+1
            BLOCK SIZE+1
BUF3:       0                                 ;THIRD BUFFER
            XWD SIZE+1,BUF1+1                 ;RIGHT HALF CLOSES THE RING
            BLOCK SIZE+1
```

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
```

CHAPTER 5

ERROR DIAGNOSTICS

MONITOR DETECTED ERRORS

Whenever a job commits an error so serious that it must be stopped, the monitor types the following preamble followed by an appropriate message to describe the fault:

ERROR IN JOB N

where N is the job number assigned to the offending job. Following the typing of an error message on the next line, the monitor returns the console to monitor mode.

In certain circumstances, such as misuse of the computer console EXAMINE switch to examine nonexistent memory, an error occurs while no job is running. In this case the monitor types:

ERROR IN JOB 0

The operator's console is the first console that received typein when the monitor was started (see Appendix 4). If the error cannot be explained by someone's misuse of the manual keys, the fault is either a monitor bug or hardware failure and should be brought to the attention of Digital personnel.

ILLEGAL MEMORY REFERENCES

If the user's program references memory above its core allotment, a hardware-initiated trap to the monitor occurs. The exec then stops the user's program and types the following message on the user's console:

ILL MEM REF AT USER LOC N

where N is the address of the offending instruction.

If the user's program attempts to jump to an address above its core memory assignment, the message becomes:

PC EXCEEDS MEM BOUND AT USER LOC M

where M is the address to which the user's program jumped.

The following message occurs only through hardware failure or a bug in the Time-Sharing Monitor. The non-existent memory trap causes the monitor to stop the user's job and type the following message on the user's console:

NON EX MEM AT EXEC LOC N

or

NON EX MEM AT USER LOC N

depending on whether the trap occurred from the user's program or from the exec. The repeated occurrence of this message should be reported to Digital personnel. The recommended recovery procedure is to examine the memory indicator lamps for memories that are hung up. Clear any such memories and restart the monitor. If no such memories are discovered, reload and restart the monitor.

If the user's program requests the monitor to perform I/O into or from any location above the user's core allotment, in the protected portion of the job data area, or (during interrupts) in the accumulators, the monitor stops the user's program and types the following message on the user's console:

ADDRESS CHECK FOR DEVICE X; EXEC CALLED FROM N

where X is the physical device name and N is the address of the offending prog. op. This error message frequently occurs when the user does not have enough core to satisfy all INBUF and OUTBUF programmed operators. If the improper memory reference occurs during an interrupt, the form of the message is slightly different:

ADDRESS CHECK FOR DEVICE X DURING INTERRUPT

This message frequently occurs when the user's program (via a program bug) destroys the ring buffer structure so that one buffer in the ring points outside the user's core to the "next" buffer.

The difference between the ADDRESS CHECK message and the ILL MEM REF message is that the former is detected by subroutines in the exec while the latter is detected by hardware.

ILLEGAL INSTRUCTIONS

Illegal instructions are all I/O instructions; prog. op. 0 and all undefined prog. ops. above 40; and JRST 10, or JRST 4, . All these instructions trap to the exec which stops the user's program and types the following message on the user's console:

ILL PROG. OP. USED AT USER LOC N

where N is the address of the offending prog. op., or:

ILL INST AT USER LOC N

depending on whether the offending instruction was a prog. op. or was a JRST or I/O instruction respectively.

OUTPUT TO AN INPUT DEVICE AND VICE VERSA

If the user's program attempts output to an input-only device or vice versa, upon the occurrence of the faulty INPUT or OUTPUT prog. op. the monitor stops the user's program and types the following error message on the user's console:

```
INPUT DEVICE X CANNOT DO OUTPUT; EXEC CALLED FROM USER LOC N
```

or

```
OUTPUT DEVICE X CANNOT DO INPUT; EXEC CALLED FROM USER LOC N
```

where X is the physical name of the device involved and N is the address of the offending prog. op.

MONITOR ERROR

If an undeterminable error occurs, either through a bug in the monitor or hardware failure, the monitor stops the user's program that is running at the time of the failure and types the following message on that user's console:

```
ERROR IN MONITOR AT EXEC LOC N
```

This message should be brought to the attention of Digital maintenance personnel. The recovery procedure is to reload the monitor and restart.

CHAPTER 6

THE STANDARD CUSP LIBRARY

THE CONTENTS OF THE CUSP TAPE

The following is an enumeration of the files found on a standard CUSP tape. Only those programs that are not described in separate manuals are covered below in complete detail. Table 6-1 contains a summary of the commands and error messages relevant to the most commonly used CUSPs.

The following programs are found on a standard CUSP tape. Those labeled "Dump" are loaded by the GET console command. Those labeled REL (relocatable binary) are loaded by the binary program loader which is itself a dump program.

Program Name	Type	Full Name	Size
9KF2	Dump	FORTRAN Compiler (two-pass version)	9K
FOLA	Dump	FORTRAN On-Line Assembler (for two-pass FORTRAN)	9K
F2	Dump	FORTRAN II (one-pass version)	22K
MACRO	Dump	MACRO Assembler	9K
PIP1	Dump	Peripheral Interchange Program (1K version)	1K or more
PIP2	Dump	Peripheral Interchange Program (2K version)	2K or more
EDITOR	Dump	DECtape Editor	1K
LOADER	Dump	Relocatable Binary Program Loader	1K plus object program size plus I/O buffers for storage map
DESK	Dump	Desk Calculator	2K
DDT	Rel	Digital Debugging Technique (on-line debugger)	1.5K plus symbol table
JOBDAT	Rel	Job Data Area Definitions	0 (defines external symbols only)
FORLIB	Rel	FORTRAN operating system and subroutine library	Up to 2.7K

Program Name	Type	Full Name	Size
FUDGE	Dump	File Updating Generator	2K or more
RIM DTA	Dump	RIM Paper Tape to DECtape Dump Conversion	1K plus the size of the program being copied

GENERAL CUSP COMMAND FORMAT

Many of the CUSPs produce output and accept input from all devices. These CUSPs (FORTRAN, FOLA, PIP, LOADER) require that the user type in the names of the devices he wishes to use. The typein formats for these CUSPs are similar. In general the format is:

X,X←X,X,X ...↵ or ALTMODE

where each "X" represents a file descriptor as illustrated below. The files to the left of the left arrow are destination (output) files; those to the right are source (input) files. The file descriptors have the following format:

device:filename.extension

where "device" is the logical or physical name of the desired device, "filename" is a 1- to 6-character file name, and "extension" is a 1- to 3-character file name extension.

For the convenience of the user, the command format may be abbreviated as follows. The file name extension need not be typed, in which case the period should also be omitted. For devices other than DECtape, no file name is required, in which case only a device name followed by a colon must be typed. If the device name is omitted, the most recently typed device name is used. Thus to specify more than one file from the same device, the user types:

PTR:,,	(three paper tape files)
PTR:,	(two files)
PTR:	(one file)
DTA3:file1,file2,file3	(three DECtape files)

When a CUSP requires more than one output file, the first file on the left is the binary file (or the intermediate file in the case of two-pass FORTRAN) and the second file is the listing file.

Most CUSPs allow the user to select several options or modes and to perform certain device-manipulation functions (such as rewinding magnetic tape). These options, called switches, are selected by typing a single letter preceded by a slash or by typing several letters enclosed in parentheses. Each letter selects an option as determined by the particular CUSP being used. For example, to command PIP to copy a binary paper tape, the user types:

```
PTP;/B← PTR:
```

where "B" selects binary mode for PIP. To copy a card file, the user might type:

```
DTA2:CARDS(CS) ←CDR:
```

where "CS" selects both Card input format and the insertion of Sequence numbers.

PERIPHERAL INTERCHANGE PROGRAM

PIP1 and PIP2 are programs to transfer data from one I/O medium to another. PIP performs simple editing functions during data transmission.

Size: PIP1=1K or more, PIP2=2K or more (both use all available storage).
Equipment: Teletype for control, other devices as desired.
Loading: Run as a user program.

Operating Instructions

Commands to PIP are in the standard CUSP format consisting of one output file and any number of input files. Nonstandard formats and dump files cannot be copied with PIP.

PIP2 accepts the switches specified below. PIP1 accepts only the switches not marked with an asterisk. Switches may appear in the command line anywhere to the left of the first comma (or anywhere if a comma is not required).

If switches are not typed, PIP transmits ASCII data (including sequence numbers if present) from each successive source to the destination, changing only the character following a sequence number to be either a tab or a carriage return. Thus, a binary file cannot be copied without the B switch.

Switch	Action
A	ASCII line mode processing. (Do not break up lines between buffers.)
B	Process binary data.
C*	Suppress trailing spaces at the end of each line and change multiple blanks into tabs. (Useful for card input.)
D*	Delete all source file names from the directory of the destination device. The destination device name supersedes any other device name that is typed.

Switch	Action
F	FORTRAN format, one line per record.
L*	List the directory of the source device. Only one source may be specified.
N	Remove sequence numbers.
R*	Rename the source file to have the name of the destination file. No source device name is necessary because the destination device is implied. Only one source may be specified.
S	A sequence number starting with 00010 and incrementing by 10 is added to the front of each line. A tab is inserted after the sequence number unless the line consists of form-feed, vertical tab, or line-feed, in which case a carriage-return is inserted. If a file is already sequenced, the old sequence numbers are removed. Sequence numbers are never split between words and are identified by a one appearing in bit 35.
T*	The same as C, except do not replace multiple blanks with tabs.
W	Rewind magnetic tape. Source files are ignored.
X*	Copy an entire DECtape, recovering unused blocks. Empty files are not copied. Does not disturb data previously written on the destination tape unless file names conflict. Although the source device must be DECtape, the destination may be any output device.
Z	Clear the destination DECtape and create a fresh directory. Sources are ignored.

When PIP has finished copying a file, it types a carriage-return/line-feed and is ready for the next command.

Some of the error messages that PIP might type out are:

DECTAPE INPUT REQUIRED	A directory manipulation command (L,R,D) was given for a device other than DECtape.
USE PIP2	One of the * commands was given to PIP1.

Examples: PIP1 and PIP2

To copy three paper tapes onto one DECtape file, type:

```
DTA1:BIGFIL ← PTR:,,
```

To do the same as above, but with inserting sequence numbers, type:

```
DTA1:BIGFIL/S ← PTR:,,
```

To list a file, type:

```
LPT: ← DTA1:BIGFIL
```

To do the same as above, but stripping off seauence numbers, type:

```
LPT:/N← DTA1:BIGFIL
```

To rewind magnetic tape:

```
MTA1:/W←
```

To copy binary data:

```
DTA1:PIP2.REL/B← PTR:
```

To create a fresh directory:

```
DTA1:/Z←
```

PIP2 Examples:

To clean up a DECtape and recover unused blocks, mount the source tape on unit 1 and a scratch tape on unit 2 (for example). Then type:

```
DTA2:/Z←
DTA2:/X← DAT1:
```

Return your old tape to the scratch tape supply and take possession of the tape on unit 2.

To rename files, type:

```
DTA2NEWNAM.EXT/R← OLDNAM
DTA2:AFTER/R    BEFORE
```

To delete files, type:

```
DTA2:/D← FILE1,FILE2,FILE3
```

To list a directory:

```
TTY:/L← DTA2:
```

FORTRAN AND FOLA

FORTRAN II is implemented in two versions, a 9K version that produces an intermediate, assembly-language file which must be assembled in a second pass by FOLA (FORTRAN On-Line Assembler), and a 22K version that contains FOLA within itself and produces relocatable binary output in one pass. The 22K version accepts all the commands that the 9K version accepts and all the commands that FOLA accepts.

Size: 9KF2=9K (F2=22K) (FOLA=9K)

Equipment: Teletype for control, other devices as desired.

Loading: Run as a user program.

Operating Instructions

For a description of the FORTRAN II language, see programming manual DEC-6-0-TP-FII-LM-FP-ACT00. The user may specify up to two destination files and any number of source files. File names are in the standard format. In general, the command format is:

For 9KF2: Intermediate File, Intermediate Listing ← Sources

For F2: Binary File, Listing ← Sources

For FOLA: Binary File, Listing ← Intermediate File

For a listing file only, use the form:

,Listing ← Sources

For no listing, use:

Binary File, ← Sources

To scan for language errors but to produce no output, use:

, ← Sources

Switches

There are two switches recognized by FORTRAN (marked by *), three by FOLA (marked by **), and three recognized by both. All eight are recognized by 22KF2.

Switch	Action
T*	The source is in paper-tape image format.
M**	Do not print a symbol map after compilation.
Z	Clear the directory on the device before inserting the new file.
R*	The source file is a set of FORTRAN II rules instead of source statements.
K	Skip a file before compilation.
W	Rewind before compilation.
N**	Do not put errors on TTY is there is a listing file.
S**	List source only.

The K and W switches are for mag-tape only.

See the PDP-6 FORTRAN II Manual for the differences between card and paper-tape image formats. To facilitate using card image format with TTY or PTR input, a tab in columns 1-6 will cause a skip to column 7. No colon should appear in columns 1-6.

More than one switch may be enclosed within parentheses; a single switch may appear after a forward slash. Any ASCII characters other than the above recognized switches will be ignored.

The switches N, S, T, M, and R may appear anywhere in the command string before the terminating carriage-return. If switch K, W, or Z is desired, it must appear before the terminating character (comma, left arrow, or carriage-return) for the desired file. The Z switch is ignored if it appears after the left arrow.

FORTRAN is normally dumped with a complete set of syntax rules preloaded. The rules need not be read in again unless they are changed. If they are to be changed, the compiler need not be reloaded; instead "GET" the previous version, and type in the following command string to read in rules:

```
(R)DTA1:RULES
```

(assuming the rules are on DTA1 in a file called "RULES"). The compiler will terminate by calling exit; "EXIT" will be typed out. The compiler may then be saved. The writing of syntax rules is described in the document, Writing Syntax Rules, DEC-6-00-TP-SC-UM-IP-PRE2. FORTRAN types "EXIT" when compilation is complete. The lack of any messages to the contrary indicates there were no errors. FORTRAN must be reloaded from the CUSP tape to perform additional compilations.

MACRO

MACRO is an assembly program featuring a powerful macro-instruction processor and relocatable assembly with automatic linking of externally defined symbols at load time.

Size: 9K (a 13K version is available by special request)
Equipment: Teletype for control, other devices as desired.
Loading: Run as a user program.

Operating Instructions

For a description of the MACRO assembly language, see programming manual DEC-6-0-TP-MAC-LM-FP-ACT01. The user may specify up to two destination files and any number of source files. File names are in the standard format. In general, the command format is:

```
Binary File,Listing ← Sources
```

For a listing without the binary output type:

```
,Listing ← Sources
```

For binary with no listing file type:

Binary File, ← Sources

To scan for language errors but to produce no output type:

, ← Sources

Switches

MACRO recognizes the following switches:

Switch	Action
S	Suppress the listing.
L	Reinstate the listing.
E	List macro-instruction expansions.
X	Stop listing macro-instruction expansions.
F	Fix the symbol table so that all symbols currently defined become initial symbols.
N	No error listings on the console.
A	Advance a magnetic tape forward one file.
B	Backspace a magnetic tape one file.
T	Skip to the logical end of a magnetic tape.
W	Rewind magnetic tape.
Z	Clear a DECtape directory.

The last five switches must be typed immediately following the device specification to which they apply.

MACRO terminates reading a source file when it encounters the pseudo-instruction TAPE or END. If an END statement does not appear before the end of the last source file typed in, MACRO types out the message:

NO END STATEMENT ENCOUNTERED ON INPUT FILE

Note that a TAPE pseudo-instruction preceding an END pseudo-instruction in the source file hides the END.

When the assembly is complete, MACRO types:

END OF ASSEMBLY

and is ready to assemble another program.

EDITOR

The EDITOR allows the user to make modifications to an ASCII sequenced file on DECtape. Programming Manual, DEC-06-TP-EDIT-LM-FP-ACT01, describes the operating procedures for the EDITOR.

Size: 1K
Equipment: Teletype for control, one DECtape transport for the edited file.
Loading: Run as a user program.

LOADER

The LOADER allows the user to load binary relocatable programs produced either by MACRO or FORTRAN (FOLA). The LOADER links externally defined symbols to the proper addresses, building a table of such symbols. This table is available to DDT for user's on-line debugging.

Size: 1K plus the size of the loaded program plus the size of the required symbol table.
Equipment: Teletype for control, other devices as desired.
Loading: Run as a user program.

Operating Instructions

The general form for commands to the loader is:

Storage Map File ← Sources

If the storage map file is never specified and the user requests a storage map, it is typed on the Teletype console.

File specifications are in standard format; however, switches are not in standard format. The characters (and / precede switches as in the standard form, but are equivalent to one another. The list of switches is terminated by any nonalphanumeric character. Moreover, certain switches have the same ability to separate file descriptors as does comma; these are marked below with an asterisk.

Switches

The following switches are accepted by the LOADER:

Switch	Action
A	List all global symbols in storage map regardless of program length.
D*	Load DDT, enter load with symbols mode.

Switch	Action
F*	Library search FORLIB, exit from load with symbols mode.
G*	Terminate loading, begin automatic library search of FORLIB, list undefined globals. An octal number may be typed preceding the "G". The numeric value is used as the starting address (if nonzero). The rest of the command line is ignored.
I	Set LOADER to ignore starting addresses in binary input.
J	Set LOADER to accept starting addresses in binary input.
L	Enter library search mode.
M*	Print storage map and undefined list.
N	Leave library search mode.
P	Prevent automatic library search.
Q	Allow automatic library search.
S	Load with local symbols.
U*	List undefined globals on destination file.
W	Load without local symbols.
X	Suppress listing global symbols in zero length programs.
Z	Restart loading. Core cleared, loader set to initial state. The rest of the command line is ignored.

A command string to the loader may consist of one or more lines. A line is terminated by typing carriage-return or alt mode. The loader signals readiness to accept one line of console input by typing an asterisk at the beginning of the line. Relocatable binary files are loaded in the order the source specifications occur. All loading required by one line of a command string occurs after the carriage-return is input, but before the asterisk giving the go-ahead to type the next line is output.

The storage map file may be specified at the beginning of any command line.

The character ALTMOD has the same effect as the G switch except that no starting address may be typed. When the loader is finished, the message:

LOADER FINISHED

is typed on the user's console and the console is placed in monitor mode. The user then has the option to dump his program using the SAVE monitor command or to start his program by typing START.

DESK

The DESK calculator performs numeric calculations on-line. It features single-precision floating point arithmetic, built-in trigonometric and logarithmic functions, and the ability to define functions. Programming Manual, DEC-06-UP-DESK-UM-FP-ACT01, describes the operating instructions for DESK.

Size: 2K

Equipment: Teletype for control and for results.

Loading: Run as a user program.

DDT

DDT provides a powerful on-line debugging language in which the user refers to his program by symbols appearing in the MACRO language source program. The operating instructions for DDT are found in Programming Manual DEC-6-0-UP-DDT-UM-FP-ACT00.

Size: 2K plus the required symbol table.

Equipment: Teletype.

Loading: Load DDT with the LOADER at the same time the program to be debugged is loaded. If DDT is to be loaded first, use the D switch to the LOADER as described above. Otherwise, be sure to load with local symbols (using the S switch) for those programs which are to be debugged. A suitable command to the Loader for loading DDT is either:

```
                         SYS:DDT
or
                           /D↵
```

The file JOBDAT is included within the DDT file and need not be loaded separately if DDT has been loaded.

> NOTE: The DDT file produces about three pages of external symbols on a storage map. It is wise to order the loading process so that all required storage maps are obtained before DDT is loaded.

JOBDAT

The JOBDAT file contains the symbolic definitions of addresses in the job data area. Users' programs that refer to the job data area should have all such locations declared as external symbols which are then defined by the JOBDAT file at load time.

Size: 0

Equipment: none

Loading: Load with the LOADER by typing:

```
SYS:JOBDAT
```

FORLIB

The file FORLIB contains the FORTRAN operating system and the FORTRAN library of standard functions in Linking Loader format. The operating system is described in the document, FORTRAN II Format and I/O Processor, DEC-06-OS-FII-GM-FP-ACT00. The arithmetic functions are described in the FORTRAN Langauge Manual, DEC-6-0-TP-FII-LM-FP-ACT00.

Size: Between 1K and 2K depending on how much of the library is required.

Equipment: Teletype for error messages and any desired devices.

Loading: Load by the Linking Loader in library search mode (L switch) after all programs requiring the FORTRAN library have been loaded. The loader automatically loads FORLIB from device SYS if undefined external symbols remain when the command to terminate loading is received. However, the user may wish to load FORLIB followed by, say, DDT in which case a suitable command to the LOADER is:

```
SYS:FORLIB/L,/D
```

FUDGE

FUDGE is a program for editing relocatable binary files, such as FORLIB, which consist of several independent programs collected into a single file. The operating instructions for FUDGE are given in the document, Description and Operating Instructions File Updating Generator (FUDGE), DEC-06-0-UP-LIB-UM-FP-ACT00.

Size: 2K or more

Equipment: Teletype for control, any three additional devices, one to receive the edited file, one to supply the old master file, one to supply the replacement programs.

Loading: Run as a user program.

RIMDTA

RIMDTA is a user program to read paper tapes in RIM format (see the MACRO manual) and to write out an equivalent dump file onto DECtape. The principal use of this program is to read maintenance tapes onto a single DECtape for convenient handling.

Size: 1K plus the size of the RIM object program.

Equipment: Teletype for control, paper tape reader for reading the RIM tape, any output device for the output files.

Loading: Run as a user program.

Operating Instructions

Only one file is specified to RIMDTA, the output file. Input from the paper tape reader (PTR) is assumed. The output file descriptor is in standard format; no commas, left arrows, or switches are required. The user may specify a date in the following format which appears in the directory for each file.

DEV:FILNAM.EXT(month/day/year)

"Month", "day", and "year" are 2-digit decimal numbers. If the file name extension is not specified, "DMP" is automatically supplied.

When RIMDTA is done, it types out either:

DONE

or

TRANSMISSION ERROR.

"DONE" means that the operation is successful and RIMDTA is ready for the next tape. "TRANSMISSION ERROR" means that the attempt to write on DECtape is unsuccessful (often because the WRITELOCK switch is in the locked position).

Files generated by RIMDTA are loaded and run in exec mode by the block 0 loader described in Appendix 4.

TABLE 6-1 SUMMARY OF COMMON USER SERVICE PROGRAMS

Program	Size	Number of DEST. Files	Switches	Commands	Error Messages
EDITOR	1K	EDITOR operates upon one file (sequenced.)	There are no switches in the EDITOR Special delimitors: CR(CARRIAGE RETURN)=Ends a command. .(POINT)=Has the value of the last line number typed. ALT MODE=Special delineator for S; when used alone advances and prints the line.	*Typed when ready for a command. Sx control A (CR)=Select unit x and clear the directory. Sx, name control A ALT MODE=Same as above but creates file "NAME". Sx, name (CR)=Open file "NAME" on unit x. Sx, name ALT MODE=Create a new file called "NAME" on unit x. E(CR)=CLOSE THE CURRENT FILE. In1(CR)=Insert at line n1. In1, I2(CR)=As above, but set indexing increment to I2. Dn1(CR)=Delete line n1. Dn1, n2 (CR)=Delete line in n1 through n2. Pn1(CR)=Print line n1. Pn1,n2(CR)=Print lines n1 through n2. Rx(CR)=Resequence the current file. Rx,y(CR)=Resequence, stepping by y. ALT MODE=Return from text mode to command mode, or if in command mode, prints the next line.	"*ILC*" Illegal command. "*NLN*" A or D referenced a nonexistent line. "*ILS*" The index step (in insert) has caused a line already in the file to be skipped. "*ILR*" The index step has caused an already existing line to be referenced. "*UNA*" DECtape unit is not available. "*DCE*" DECtape directory full. "*NFO*" No file has been selected. "*FAU*" File name already in use. "*NCF*" Not a current file. "*DDE*" Device data error or write lock.
PIP1	1K Minimum (Uses all available storage)	1	A=ASCII line. B=Binary. F=FORTRAN format, one line per record. N=Delete sequence numbers. S=Sequence. W=Rewing mag tape. Z=Create fresh directory.	General Form: Destination ⟵ Source 1. Source 2, ... DTA1:Big Fil ⟵ PTR:,, (Copy 3 paper tape files onto DECtape). DTA1:Big Fil/S ⟵ PTR:,, (Same as above but add sequence no.). LPT: ⟵ DTA1:Big Fil (List the file). MTA1/W ⟵ (Rewind mag tape). DTA2:/Z ⟵ (Create a fresh directory).	"DEVICE name NOT AVAILABLE" "DIRECTORY FULL" "NO FILE NAMED file name" "OUTPUT DEVICE name DEVICE ERROR" "OUTPUT DEVICE name DATA ERROR" "OUTPUT DEVICE name IMPROPER MODE" "OUTPUT DEVICE name BLOCK TOO LARGE" "USE PIP2" "INPUT FILE file name DEVICE ERROR" etc.

TABLE 6-1 SUMMARY OF COMMON USER SERVICE PROGRAMS (continued)

Program	Size	Number of DEST. Files	Switches	Commands	Error Messages
PIP2	2K Minimum (Uses all available storage)	1	A=ASCII line. B=Binary. C=Card input suppress trailing spaces. Imbedded multiple spaces to tabs. D=Delete a file. F=FORTRAN format. L=List directory of source. N=Delete sequence numbers. R=Rename source file with name of dest. S=Add new sequence numbers. W=Rewind mag tape. X=Copy and clean up DECtape. Z=Create a fresh directory. T=Same as C, but without tabs.	General Form: Same as PIP1, but in Addition: DTA2:/X ⟵ DTA1: (Copy and clean up). DTA2:After/R ⟵ Before (Rename A file). DTA2:/D ⟵ File 1, File 2 (Delete two files). LPT:/L ⟵ DTA2: (List the directory).	Same as PIP1 and "DECTAPE INPUT REQUIRED"
FOLA	5K	2	N=No error listings on the console. S=List source only. M=Don't print symbol table. W=Rewind mag tape. Z=Clear DECtape directory. K=Advance forward mag tape one file.	General Form: Binary, Listing ⟵ Source 1, Source 2, ... PTP:, LPT: ⟵ PTR: DTA2: BINARY, DTA3: LIST ⟵ DTA1: FOLAIN ⟵ DTA3: FOLAIN (No output, errors only). PTP: ⟵ PTR: (Binary onto paper tape, no listing). , LPT: ⟵ PTR: (No binary, listing on printer).	"??"=Command error "device name UNAVAILABLE" "DEVICE ERROR ON INPUT" "DATA ERROR ON INPUT" "IMPROPER INPUT MODE ON device name" "INPUT BLOCK TOO LARGE ON device name" "OUTPUT ERROR" "device name DIRECTORY FULL" "file name NOT IN DIRECTORY" "device name IS NOT A DEVICE" "device name CANNOT DO BINARY" "NO END STATEMENT ON FINAL PROGRAM"

TABLE 6-1 SUMMARY OF COMMON USER SERVICE PROGRAMS (continued)

Program	Size	Number of DEST. Files	Switches	Commands	Error Messages
FORTRAN	9K or 22K	2	T=Teletype or paper tape input. R=Read in rules (paper tape image assumed). For 22K version, all FOLA switches are recognized. For 9K version, only W, Z, and K.	22K compiler: Binary, Listing ⟵ Source Ex: DTA3:BINFIL, LPT: ⟵ DTA2:FORTIN ⟵ PTR:(T) (No output, errors only). 9K Compiler: Fola Input, Listing ⟵ Source Ex: DTA2:FOLAIN, DTA1: LIST ⟵ DTA5: FORTIN (Card image). DTA3:FOLAIN ⟵ PTR: (No listing, paper tape input). DTA3:FOLAIN ⟵ DTA1:PIMAGE(T) (Teletype image).	Same as FOLA.
MACRO	9K	2	S=Suppress listing. L=Reinstate listing. E=Expand macros on listing. X=Turn off expansion F=Fix symbol table. N=No error listings on the console. A=Advance forward mag tape one file. B=Backspace mag tape one file. T=Skip to logical end of tape. W=Rewind mag tape. Z=Clear DECtape directory.	General Form: Binary, Listing ⟵ Source 1, Source 2, ... PTP:, LPT:, ⟵ PTR:,,, (4 Input files). ⟵ DTA3: Macin (No output, errors on user TTY). DTA1: Bin Out ⟵ DTA2: Macin (No listing errors on TTY). , LPT: ⟵ MTA1: Assemb (No binary). DTA2: Bin Out, LPT: ⟵ (S)DTA6:MAC1, (L)PTR:, (No listing of file MAC1, two files from PTR).	"INPUT ERROR ON DEVICE X" "DATA ERROR ON DEVICE X" "NO END STATEMENT ENCOUNTERED ON INPUT FILE" "CANNOT FIND file name"

TABLE 6-1 SUMMARY OF COMMON USER SERVICE PROGRAMS (continued)

Program	Size	Number of DEST.Files	Switches	Commands	Errors Messages
LOADER	1K	1	S=Load local symbols. W=Do not load local symbols. L=Enter library search mode. N=Leave library search mode. I=Ignore all subsequent starting addresses. M=Print storage map. Z=Restart loading. ALT MOD=Finish loading. Enter L mode and load FORLIB if any global symbols are undefined. A=List all globals including those in zero length programs. D=Load DDT from SYS Tape. F=Leave S mode, enter L mode and load FORLIB from SYS Tape. Number G=Same as ALT MOD. If number #0, it is the starting address. J=Opposite of I. P=Prevent automatic library search for ALT MOD. Q=Opposite of P. U=List undefined globals only. X=Suppress listing globals in zero length programs. Normal mode is X, W, N, Q, J.	General Form: Memory Map ← Switches, Source 1, ... LPT: ← DTA3:BIN1(S),PTR:,DTA4:USRDDT (S)PTR: (Map on user TTY).	"ILL FORMAT file name" "device UNAVAILABLE" "INPUT ERROR file name" "ILL COMMON file name" "EXIT"= not enough core "char CHAR.ERROR IN LOADER COMMAND" "char SYNTAX ERROR IN LOADER COMMAND" "char SWITCH ERROR IN LOADER COMMAND" "n WORDS OF OVERLAP"= n words of extra core required "symbol n m MUL.DEF.GLOBAL file name" where n is the ignored value and m is the real value. "CANNOT FIND file name" "DIR FULL"

APPENDIX 1

MULTIPROGRAMMING EXECUTIVE COMPONENTS

This appendix contains a brief functional description of each of the components of the Multiprogramming Monitor. This section is intended to convey an idea of the size, function, and organization of the monitor, but not the complete details on methods of operation. Components sizes are subject to change without notice.

DIRECT JOB CONTROL ROUTINES

These routines are directly associated with processing a user's job. They handle the console communications, job scheduling, etc.

Program Name	Size (octal)	Function
SYSCON	252	Processes UUOs and illegal instructions. Dispatches to the appropriate subroutine for each UUO. Returns to the user for user's UUOs.
COMINI	135	Interprets console commands. Dispatches to system subroutines to perform the actions requested.
CORE	100	Processes all core allocation requests.
CLOCK	144	Handles processor interrupts. Calls the scheduling algorithm to schedule a new job. Dispatches to error routines for job error traps. Processes the clock queue.
RUNCSS	270	Controls the actual starting and stopping of jobs.
ERRCON	425	Transmits error messages to the user's console. Interprets error conditions.
IOCONT	477	Processes the standard input/output UUOs. SYSCON dispatches to this program.

EXECUTIVE DATA STORAGE

These routines provide data storage tables for the various system functions. These storage tables define the system configuration and (together with the accumulators) the state of the monitor program.

Program Name	Size (octal)	Function
IOINI1	443	Contains all system tables and temporary registers except those for the I/O modules. Also contains the priority interrupt processing subroutines.
IOINI2	2636 (DEC config)	Contains all tables for the I/O subroutine modules. The size of this program is highly dependent on system configuration.
JOBDAT	0	Defines the names of locations in the user's job data area.

INTERNAL EXECUTIVE SUBROUTINES

These programs contain subroutines and other useful functions that are called from other system programs.

Program Name	Size (octal)	Function
CLKCSS	102	Performs the scheduling algorithm.
IOCSS	401	Contains subroutines that are used throughout the I/O service modules.
SYSCSS	35	Processes the miscellaneous system UUOs.
COMCON	272	Processes console commands. COMINI dispatches to this program.
COMCSS	234	Contains subroutines used in processing console commands.
SAVGET	164	Processes the two console commands, SAVE and GET.

SPECIAL PROGRAMS

The following programs perform system initialization and system debugging. They are not normally referenced during time-shared operations.

Program Name	Size (octal)	Function
SYSINI	110	Initializes executive storage areas. Processes manual restarts. Initializes I/O modules.
ONCE	563	Processes the initial conversation in which the operator types in the day and the time.
SYSMAK	36	Moves the core image for job number one down on top of the monitor to the corresponding absolute locations. This program permits nonuser programs to be loaded by the time-sharing loader and to be run in exec mode.
DDT	3203	The executive debugging program. The space SYSMAK and DDT occupy may be used as additional user's program area.

Program Name	Size (octal)	Function

I/O SERVICE MODULES

The following programs process I/O service requests and device interrupts for the various devices that may be used during time-sharing.

Program Name	Size (octal)	Function
SCNSER	1163	Controls the console Teletype and up to 64 lines on the Data Communications System Type 630, both full and half duplex. Processes console communications with the monitor.
DTSER2	1305	Controls up to eight DECtape transports on one DECtape Control Type 551. Interlocks with the magnetic tape service module for the use of the data control.
LPTSER	141	Controls one Line Printer Type 646.
CDRSER	244	Controls one Card Reader Type 461.
PTPSER	231	Controls one Paper Tape Punch Type 761.
PTRSER	140	Controls one Paper Tape Reader Type 760.
MTPSER	551	Controls up to eight magnetic tape transports on one Magnetic Tape Control Type 516. Interlocks with the DECtape service module for the use of the data control.

APPENDIX 2

MONITOR SCHEDULING ALGORITHM

The objective of the scheduling algorithm is to keep the I/O devices as busy as possible while still giving users good response for short computations. To do this the monitor keeps track of which jobs are runable and which are not. At least every 17 msec the monitor reviews the situation and decides whether or not to continue running the current job. This is done by keeping the runable jobs in a queue and serving them in a round robin manner with the head of the queue always being run. Whenever an unrunable job becomes runable it is placed at the front of the queue and is assigned a fixed quantum of time depending on the reason it became runable. It will then be run at the next scheduling period which can never be more than 17 msec away. Whenever the head of queue job exceeds its quantum, it is reassigned a somewhat longer quantum of 250 msec and is placed at the end of the queue.

Jobs may become runable for any of the following reasons:

1. The user typed a console command which started execution such as START, CONT, STARTC, CONTC, or DDT. He is given a 100-msec quantum.

2. The job just completed waiting for some of its own input/output to finish. It is given a 100-msec quantum.

3. The job just completed waiting for some other job's input/output to finish on a device which must be shared in a serial manner such as the magnetic tape control, the DECtape control, or the data control. The job is given a 70-msec quantum.

The effectiveness of a time-sharing system depends to a great extent on the program scheduling strategy used. Realizing this, the entire scheduling algorithm has been incorporated into a single separate subroutine. Thus, it may be changed easily to meet the needs of a particular computer installation.

The program name of this subroutine is CLKCSS and its only entry point is NXTJOB. Its sole function is to return the job number of the highest priority job at that instant. If no job is runable, it returns 0, the so-called null job which counts in AC0 and runs with the PC at 1. NXTJOB is called on the following occasions:

1. Every 60th of a second.

2. Whenever the current job is forced to wait for I/O or is stopped by the user at his console ('control' C).

3. Whenever any job becomes runable while the null job is running.

For further details consult the assembly listing of the subroutine itself.

APPENDIX 3

SYMBOL DEFINITIONS

OPDEF TAPE FOR EXEC SYSTEM

OPDEF ENTER	[77B8]
OPDEF LOOKUP	[76B8]
OPDEF USETO	[75B8]
OPDEF USETI	[74B8]
OPDEF UGETF	[73B8]
OPDEF MTAPE	[72B8]
OPDEF RELEAS	[71B8]
OPDEF CLOSE	[70B8]
OPDEF OUTPUT	[67B8]
OPDEF INPUT	[66B8]
OPDEF OUTBUF	[65B8]
OPDEF INBUF	[64B8]
OPDEF STATZ	[63B8]
*OPDEF STATUS	[62B8]
OPDEF GETSTS	[62B8]
OPDEF STATO	[61B8]
OPDEF SETSTS	[60B8]
OPDEF INIT	[41B8]
OPDEF CALL	[40B8]

OPDEF'S FOR FORTRAN OPERATIONS SYSTEM

OPDEF RESET.	[15B8]
OPDEF IN.	[16B8]
OPDEF OUT.	[17B8]
OPDEF DATA.	[20B8]
OPDEF FIN.	[21B8]
OPDEF RTB.	[22B8]
OPDEF WTB.	[23B8]
**OPDEF REW.	[24B8]
OPDEF SLIST.	[25B8]
OPDEF INF.	[26B8]
OPDEF OUTF.	[27B8]
OPDEF RERED.	[30B8]

*Obsolete name

**The following UUOs have the same definition as REW.: REWUN., BSR., WEF., SPR., and WBR.

```
TITLE  JOBDAT - JOB DATA AREA ASSIGNMENTS
SUBTTL 8-2-65

        DEFINE BLK (A,B)
        <JOBPRO=B
        A=Ø
        ENTRY A
        DEFINE BLK (C,D)
        <C=JOBPRO
        JOBPRO=JOBPRO+D
        ENTRY C>>

        BLK JOBAC,2Ø     ;SYSTEM USE AC STORAGE
        BLK JOBDAC,2Ø    ;USER'S AC STORAGE WHILE JOB IS INACTIVE
JOBDHI=JOBDAC+17         ;HIGHEST AC SAVED WHILE INACTIVE
        BLK JOBUUO,1     ;UUO SAVE AND TRAP LOCATIONS
        BLK JOB41,1      ;UUO JSR LOCATION
        BLK JOBPC,1      ;BITS Ø-5=APR FLAGS, C(RH)=PC
        BLK JOBPDP,1     ;PUSH DOWN POINTER FOR SYSTEM USE
JOBPRT=JOBPDP            ;FIRST LOC PROTECTED FROM INTERRUPT SERVICE
JOBPR1=JOBPRT+1          ;FIRST LOC+1
        BLK JOBREL,1     ;C(LH)=Ø, C(RH)=RELOCATION ADDRESS
        BLK JOBUXT,1     ;EXIT FROM UUO AT BREAK
        BLK JOBLEV,1     ;C(4Ø) WHEN INACTIVE
        BLK JOBSAV,1     ;TEMPROARY STORAGE FOR UOO HANDLER
        BLK JOBPOV,1     ;PUSH DOWN OVERFLOW TRAP
        BLK JOBTEM,1     ;(UNUSED)TEMPORARY SYSTEM STORAGE
        BLK JOBTM1,1     ;"
        BLK JOBINF,1     ;UNUSED
        BLK JOBJDA,2Ø    ;JOB IO DEVICE CHANNEL ASSIGNMENTS
        BLK JOBDDT,1     ;ADDRESS OF USER'S DDT
JOBPFI=JOBDDT-1          ;LAST LOC PROTECTED FROM IO SERVICE
JOBPFU=JOBDDT            ;LAST LOC PROTECTED FROM USER
        BLK JOBCDP,1     ;(UNUSED)POINTER TO COMMAND STRING
        BLK JOBTRP,2Ø    ;(UNUSED)TRAP LOCATIONS
        BLK JOBSYM,1     ;SYMBOL TABLE POINTER
        BLK JOBUSY,1     ;(UNUSED)POINTER TO UNDEFINED SYMBOL TABLE
        BLK JOBSA,1      ;C(RH)=STARTING ADDRESS
                         ;C(LH)=HIGHEST LOCATION USED
        BLK JOBFF,1      ;FIRST FREE LOCATION
JOBDAT=14Ø      ;LENGTH OF JOB DATA AREA
JOBHGH=JOBDAT-1          ;HIGHEST LOC IN JOB DATA AREA
JOBPDL=JOBPRO-1          ;FIRST LOC. -1 OF PD LIST
JOBPD1=JOBPDL+1          ;FIRST LOC. IN PD LIST
MJOBPD=JOBPDL-JOBHGH     ;- LENGTH OF PD LIST
JOBS41=JOBPRO            ;PLACE WAHERE SAVE-GET SAVES USER LOC 41
JOBDA=JOBDAT
ENTRY JOBPFI,JOBPFU,JOBPDL,JOBPRO,MJOBPD,JOBHGH
ENTRY JOBDHI,JOBPRT,JOBPR1,JOBDA,JOBS41,JOBPD1

        END,
        XLIST


DEFINE  XP(A,B)             ;SYSTEM PARAMETER
        <A=B
        INTERNAL A
>
,ACCUMULATORS
        XP IOS,Ø;                   IO STATUS WORD
        XP TAC,1;                   TEMPORARY
        XP TAC1,2;                  TEMPORARY
        XP PDP,3;                   C(LH)=NUMBER OF LOCATIONS LEFT IN PD LIST
,                           C(RH)=PUSHDOWN POINTER
REPEAT Ø,<      XP SUB,1Ø                   RETURN (PC) FOR ONE LEVEL SUBROUTINES
>
        XP TEM,1Ø
        XP ITEM,4;          BYTE OR WORD POINTER.  ITEM COUNT
        XP DAT,5;                   DATA OR TEMPORARY
        XP DEVDAT,6;        C(LH)=UNUSED,
,                           C(RH)=ADDRESS OF DEVICE DATA BLOCK
        XP JBUF,DAT;        ADDRESS OF JOB BUFFER AREA=JBFADR
        XP PROG,7;                  C(LH)=UNUSED
,                           C(RH)=ADDRESS OF USER'S PROGRAM AREA
        XP JDAT,11;         C(RH)=ADDRESS OF JOB DATA
,ONLY Ø TO 1Ø NORMALLY REQUIRED
,FOR INTERRUPT SERVICE
        XP BUFPNT,12;       C(JBFADR 18-35)
        XP BUFWRD,13;       BIT Ø=IOUSE
,                           BITS 1-17=BUFFER SIZE
,                           BITS 18-35=NEXT BUFFER ADDRESS
        XP UUO,14;          LAST UUO PROCESSED
        XP AC1,15           ;TEMPORARY ACS(MORE TEMPORARY
        XP AC2,16           ;THAN TAC,TAC1)
        XP AC3,17           ;
```

```
, DEVICE DATA BLOCK NAMES
        XP DEVNAM,0;    NAME IN SIXBIT ASCII
,                       C(LH)=DEVICE MNEMONIC
,                       C(RH)=DEVICE NUMBER, LEFT JUSTIFIED
        XP DEVCHR,1;    CHARACTERISTIC
,                       BITS 0-8=JOB NUMBER
,                            ZERO VALUE IMPLIES NOT ASSIGNED
,                       BITS 9-11=PI PRIORITY CHANNEL FOR DEVICE
        XP PICHN,100    ;RIGHTMOST BIT OF PI CHAN. FIELD
,                       BIT 12=UNUSED
,                       BIT 13=IORET. 0 IF DEVICE HAS A SHORT
,                           DISPATCH TABLE.  1 IF LONG
        XP IORET,20     ;FOR RETRIEVABLE DEVICES.
                ;SHORT DISPATCH TABLE =0, LONG = 1
,                       BITS 14-17=JOB DEVICE CHANNEL NUMBER
,                       BITS 18-23=DEVICE NUMBER, BINARY
,                       BITS 24-35=BUFFER SIZE
        XP DEVIOS,2;    STATUS WORD.  SEE BELOW
        XP DEVSER,3;    C(LH)=NEXT DEVICE DATA BLOCK
,                       C(RH)=DEVICE SERVICE DISPATCH TABLE
,    DEVICE SERVICE DISPATCH TABLE ASSIGNMENTS
        XP DRL,0;                   XXXDSP   :  RELEASE
        XP DCL,1;                   XXXDSP +1:  CLOSE
        ;IMMEDIATE ADDRESS PART OF CLOSE UUO
                XP CLSOUT,1     ;INHIBIT CLOSING OUTPUT
                XP CLSIN,2      ;INHIBIT CLOSING INPUT
        XP DOU,2;                   XXDSP +2:  OUTPUT
        XP DIN,3;                   XXXDSP +3:  INPUT;SHORT DISPATCH TABLE
        XP DEN,4;                   XXXDSP +4:  ENTER
        XP DLK,5;                   XXXDSP +5:  LOOKUP
        XP DDO,6;                   XXXDSP +6:  DMPO
        XP DDI,7;                   XXXDSP +7:  DMPI
        XP DSO,10;          XXXDSP+10:  SETO
        XP DSI,11;          XXXDSP+11:  SETI
        XP DGF,12;          XXXDSP+12:  GETF;LONG DISPATCH TABLE
        XP DEVMOD,4     ;BITS 6-11=LEFT HALF OF IMAGE MODE BYTE POINTER
                        ;BIT 35-J=1 IF MODE J IS LEGAL FOR THIS DEVICE
                        ;BIT 18 DEVICE ASSIGNED BY CONSOLE COMMAND
                        ;BIT 19 DEVICE ASSIGNED BY PROGRAM (INIT)
;IRGHT HALF OF DEVICE MODE WORD
        XP ASSCON,400000        ;ASSIGNED BY CONSOLE COMMAND ASSIGN
        XP ASSPRG,200000        ;ASSIGNED BY PROGRAM(INIT UUO)
;LEFT HALF DEVICE CHARACTERISTICS(DEVCHR UUO)
        XP DVOUT,1      ;OUTPUT DEVICE
        XP DVIN,2       ;INPUT DEVICE
        XP DVDIR,4      ;HAS A DIRECTORY
        XP DVTTY,10     ;IS A TTY
        XP DVMTA,20     ;IS A MAG TAPE(REWIND)
        XP DVAVAL,40    ;1 IF DEVICE IS AVAILABLE TO THIS JOB
                        ;SET BY DEVCHR UUO
        XP TTYUSE,10000 ;TTY DDB IN USE FLAG
        XP TTYATC,20000 ;TTY ATTACHED TO JOB IF 1
        XP DVLPT,40000  ;IS A LPT (CARRIAGE CONTROL IN FORTRAN)
        XP DVCDR,100000 ;IS A CARD READER(TRAILING SPACES FOR MACRO)
        XP DVDIRIN,400000       ;DIRECTORY IN CORE IF 1

        XP DEVLOG,5;    LOGICAL NAME FOR JOB DEVICE
        XP DEVBUF,6;    C(LH)=OUTPUT BUFFER AREA DDRESS
                        C(RH)=INPUT BUFFER AREA ADDRESS (JBFADR)
        XP DEVIAD,7;    C(LH)=PROG IN INDEX FIELD
XP DEVADR,DEVIAD
                 C(RH)=CURRENT INPUT BUFFER ADDRESS
XP DEVOAD,10;    C(LH)=PROG IN INDEX FIELD
XP DEVPTR,DEVOAD
                 C(RH)=CURRENT OUTPUT BUFFER ADDRESS
XP DEVCTR,11;
```

```
, I/Ø STATUS WORD ASSIGNMENTS
,DATA MODES:  BITS 32-35
        XP A,Ø;             ASCII
        XP AL,1;                    ASCII LINE
        XP I,1Ø;                    IMAGE
        XP IB,13;           IMAGE BINARY
        XP B,14;                    BINARY
        XP DR,16                    ;DUMP BY RECORDS
        XP D,17             ;DUMP ACROSS RECORDS
, STATUS BITS
,RIGHT HALF (USER)
        XP IOWC,2Ø;         DON'T COMPUTE WORD COUNT
        XP IOCON,4Ø;        CONTINUOUS (CONT=Ø)
        XP IORDEL,1ØØ;      READ AND DELETE
        XP IONRCK,1ØØ;      READ WITH NO. REREAD CHECK
        XP IODEND,2ØØØØ;            DATA END ENCOUNTERED
        XP IODERR,2ØØØØØ;           DEVICE ERROR
        XP IODTER,1ØØØØØ;           DATA ERROR
        XP IOIMPM,4ØØØØØ;           IMPROPER MODE REQUESTED
        XP IOBKTL,4ØØØØ;            BLOCK TOO LARGE
        XP IOACT,1ØØØØ; DEVICE ACTIVE
  LEFT HALF (SYSTEM)
        XP IOW,1;                   I/O WAIT
        XP IOBEG,2;         VIRGIN DEVICE
        XP IODISC,4ØØØØØ;           DISCONNECT REQUEST
        XP IOFST,4;         NEXT ITEM WILL BE THE FIRST ITEM OF A BUFFER
        XP IOSTRT,1Ø;       IO READY TO START
        XP IO,2Ø;                   OUT=1, IN=Ø
        XP IOEND,4Ø;        SERVICE ROUTINE HAS TRANSMITTED LAST DATA


;LEFT HALF USRJDA (JOB DEVICE ASSIGNMENTS) UUO'S FOR THIS CHANNEL SINCE LAST INIT
        XP INITB,4ØØØØØ;            INIT
        XP IBUFB,2ØØØØØ;            INIT WITH INPUT BUFFER SPECIFIED
        XP OBUFB,1ØØØØØ;            INIT WITH OUTPUT BUFFER SPECIFIED
        XP LOOKB,4ØØØØ; LOOKUP
        XP ENTRB,2ØØØØ; ENTER
        XP INPB,1ØØØØ;  INPUT
        XP OUTPB,4ØØØ;  OUTPUT
        XP CLOSB,2ØØØ;  CLOSE
                                    ;RELEASE CLEARS THEM ALL
;CLOSE UUO BITS
        XP CLSOUT,1         ;INHIBIT CLOSING OUTPUT
        XP CLSIN,2          ;INHIBIT CLOSING INPUT
```

```
, JOB BUFFER AREA HEADER
        XP JBFADR,0;      BIT 0=1 IF THIS BUFFER RING HAS NEVER BEEN
,                              REFERENCED FROM THE USER'S PROGRAM BY
,                              AN INPUT OR OUTPUT COMMAND.
,                         BITS 1-17=UNUSED
,                         BITS 18-35=CURRENT BUFFER ADDRESS
        XP JBFPTR,1;      BYTE POINTER TO NEXT BYTE -1
        XP JBFCTR,2;      POSITIVE ITEM COUNT
, JOB BUFFER HEADER                 BIT 0=IOUSE
        XP IOUSE,400000;                   1 IF BUFFER IS FULL (OR BEING EMPTIED)
,                              0 IF BUFFER IS EMPTY (OR BEING FILLED)
,                         BITS 1-17=BUFFER SIZE
,                         BITS 18-35=NEXT BUFFER ADDRESS
```

```
,SYSTEM PARAMETERS
        XP USRMOD,10000;            USER MODE BIT IN ARP FLAGS, PC WORD
        XP IODONE,400000;
,JOB STATUS WORD (JBTSTS)
        XP IOWS,400000; IO WAIT SATISFIED
        XP RUN,200000;  READY TO RUN OR IS RUNNING
        XP JIOW,100000; JOB IN IO-WAIT STATE
        XP DTW,40000;   DECTAPE CONTROL WAIT
        XP DCW,20000;   DATA CONTROL WAIT
        XP MTW,10000;   MAG TAPE CONTROL WAIT
        XP JNA,4000;    JOB NUMBER ASSIGNED
        XP JERR,2000;   JOB ERROR(ILLEGAL MEM. ETC.)

        XP RUNABLE,RUN+JNA;         STATUS BIT PATTERN FOR A JOB TO STAY RUNABLE
```

```
;SPECIAL ABSOLUTE LOCATIONS IN LOWER MEMORY
;THESE ARE INTENDED TO BE EXAMINED AND CHANGED BY
;USING THE DATA SWITCHES BEFORE SYSTEM IS STARTED AFTER BEING LOADED
;ONLY TODAY AND BCDTIM NEED BE SET
;ALL OTHERS ARE ASSEMBLED IN AND SHOULD BE SET ONLY IF A CHANGE IS DESIRED

XP DDTMEM,37     ;MEMORY SIZE FOR SYSTEM DDT AND SYSMAK ONLY
XP DDTSYM,36     ;SYSTEM DDT SYSBOL TABLE POINTER (USED TO BE 77)
, SYSTEM MACROS
DEFINE   ADRCHK (A)
<        EXTERNAL ADRCK
DEFINE   ADRCHK (B)
          <PUSHJ PDP,ADRCK
          HRRZ TAC,B>
          PUSHJ PDP,ADRCK
          HRRZ TAC,A>


         LIST
```

APPENDIX 4

MONITOR OPERATING INSTRUCTIONS

PROCEDURE FOR LOADING THE MULTIPROGRAMMING MONITOR

1. Turn on the computer (POWER switch on console). This should also send power to all of the I/O devices.

2. Check to see that the power is turned on for all I/O devices.

3. Place DECtape copy of monitor on drive 0. (Set the DECtape unit selection switch to 8 to get unit 0). Make sure no other units are set to 8.

4. If machine does not have fast accumulators:

   a. Assume that the 10 instruction Read-in Mode Loader is still in locations 20 through 27. (It usually is.)

   b. Depress the READER OFF lever switch.

   c. Place a RIM paper tape copy of the DECtape RIM Loader in the reader. (See DEC 06 L DTRL UML FP ACT00 Shadow Mode Loader for a listing of this 14-instruction program.)

   d. Turn on the reader.

   e. Set the ADDRESS switches to 20 and depress the INSTRUCTION STOP, I/O RESET, and START lever switches in that order. The 10 instruction Read-in Mode Loader will load the 14-instruction DECtape RIM Loader into locations 0 through 13 and then stop.

   f. Depress the INSTRUCTION CONTINUE lever switch. The DECtape RIM Loader will then load the DECtape Block 0 Loader from block 0 of unit 0 into locations 37400-37777.

   g. If the paper tape does not move when the START lever switch is depressed, the assumption that the 10 instruction Read-in Mode Loader was intact proved to be valid. It must be keyed into locations 20 through 27 using the DATA and ADDRESS switches and the DEPOSIT lever switch. The Read-in Mode Loader is:

```
        20/     710600  000060      CONO PTR, 60
        21/     710740  000010      CONSO PTR, 10
        22/     254000  000021      JRST .-1
        23/     710440  000026      DATAI PTR, 26
        24/     710740  000010      CONSO PTR, 10
        25/     254000  000024      JRST .-1
        26/     irrelevant
        27/     254000  000021      JRST 21
```

Go back to step 4a.

5. If machine has fast accumulators:

a. Assume that the DECtape RIM Loader is still in locations 0 through 13 of shadow mode memory. (It usually is.)

b. Set the ADDRESS switches to 0, depress the INSTRUCTION STOP and lift THE START lever switch.

c. The 14-instruction DECtape RIM Loader will then load the DECtape Block 0 Loader from block 0 of unit 0.

d. If block 0 is not loaded, the assumption that the DECtape RIM Loader was intact proved invalid. To restore it to locations 0 through 13 of shadow mode memory, flip on the RIM MAINTENANCE switch in bay 2, so that the RIM SUBR light on the panel on bay 1 is on.

e. Load the DECtape RIM Loader from paper tape into locations 0 through 13. See steps 4a through 4e. Do not press INSTRUCTION CONTINUE (step f).

f. Turn the RIM MAINTENANCE switch off so that the RIM SUBR light is off. Go back to step 5a.

6. Instruct the DECtape block 0 loader to load the dump file which contains the monitor by typing the file name (name of the installation) followed by carriage-return on the console Teletype. The file directory on block 1 will be searched, and the specified dump file will be loaded into core. The dump file must be a program less than 16K. If the dump file is not in the directory, a bell will be typed to indicate the error.

7. Line-feed will be typed out after the file has been loaded. Now set the ADDRESS lever switches to 140 and depress the INSTRUCTION STOP, I/O RESET, and START lever switches in that order.

8. The monitor will then enter a once-only dialogue with the operator on the console Teletype (CTY). The monitor will request various pieces of information like the date and time of day. The operator must type in this information followed by a carriage-return. A line-feed will be echoed. Rubouts work as in time-sharing and may be used to delete erroneous typing. Typing in carriage-return while monitor is typing out will terminate the line, thereby speeding up the startup procedure.

9. The once-only dialogue looks like the following with output indicated by *:

```
*DEC 1.4 10-13-65 MONITOR JUST LOADED
*TYPE TODAY'S DATE AS ABOVE.
 (operator types date)
*TYPE 4 DIGIT TIME.
 (operator types time)
*I/O CONFIGURATION
*10 TTY'S
*8 DTA'S
 etc.
*DO YOU WANT SYSMAK (TYPE Y IF YES, CR IF NO)?
 (operator types Y or CR)
*EXEC DDT?
 (operator types Y or CR)
*EXEC IS 13573 OCTAL LOCATIONS LONG.
```

10. The monitor sets its system data storage to 0.

11. The monitor examines all 256K of memory to find which are nonexistent. A memory with the power turned off will appear nonexistent and will not be used by the monitor.

12. All I/O devices are initialized

13. The PI channels are turned on and time-sharing may begin. Whenever the monitor has nothing to do, it will run the null job. The null job counts in AC0 and runs in AC1. Therefore, it can be detected by a 1 appearing in the PROGRAM COUNTER lights while the machine is in EXEC mode. The count in AC0 may be displayed by setting the ADDRESS switches to 0. The count is reset every time the null job begins to run, so that it can be used to estimate the system idle time.

14. The operator types DEASSIGN OPR which initializes job 1 and deassigns his Teletype. The operator's Teletype is given the name OPR and is defined as the first Teletype on which a character is typed in after the PI is turned on. Usually it is CTY.

15. The operator mounts the CUSP tape on drive 0 and sets it to WRITE LOCK.

16. The operator mounts console scratch tapes on all other drives, and sets them to WRITE. Make sure that all units are dialed to unique numbers starting at 0 (8). The system will hang up if a user attempts to use a unit on which no tape is mounted or the power is off.

17. Remote users may then communicate with the operator by using PIP1 in the following manner:

```
CORE 1
GET SYS:PIP1
START
OPR: ← TTY:
the message
<control> Z
```

The message should begin with a few bell characters to attract the attention of the operator. To receive an acknowledgement the message should end with GA (go ahead). Then the user should type the following PIP command string:

```
TTY: ← OPR:
```

The operator will not need to know from which Teletype the message was typed.

This interconsole message communication is most useful to request the operator to mount a tape on a drive previously ASSIGNED to the user by the ASSIGN command. It is also useful for requesting a tape to be set to WRITE or WRITE-LOCK. Example of a typical sequence of commands for remote users. Input is indicated by underline.

```
        User's Console                         Operator's Console

                                         DEASSIGN OPR
                                         JOB 1 INITIALIZED
                                         DEC 1.4H
ASSIGN DTA
DEVICE DTA1 ASSIGNED
JOB 2 INITIALIZED
DEC 1.4H

CORE 1
```

```
User's Console                               Operator's Console

GET SYS:PIP1
JOB SETUP

START


OPR:←TTY:
<bell><bell><bell>
PLEASE MOUNT TAPE 39 ON UNIT 1
SET TO WRITE GA
↑Z
                                     PLEASE MOUNT TAPE 39 ON UNIT 1
                                     SET TO WRITE GA
TTY:←OPR:
                                     TAPE MOUNTED.
                                     ↑Z

TAPE MOUNTED.
```

DECtape Colored Label Conventions

Yellow	Console Scratch
Green	System tapes CUSP + Monitor
Red	Private tapes

ERROR RECOVERY PROCEDURES

Occasionally the system may hangup. If this happens, a software trouble report should be filled out and sent to Digital Equipment Corporation, Maynard, Mass., attention: PDP-6 Librarian. Prompt replies are sent back regardless of whether the trouble is hardware or software.

The error recovery procedure consists of restarting the monitor which will reinitialize part of itself before resuming operation. There are four restart procedures which reinitialize an increasing amount of the monitor.

1. To recover from a priority interrupt which will not dismiss (PI in PROGRESS lights on continuously):

   a. Press STOP lever switch.
   b. Fill out software trouble report.
   c. Set ADDRESS switches to 144.
   d. Press START lever switch.

This action tries to dismiss interrupt and start running null job. This affects at most one job which the user will have to restart.

2. To recover from hung I/O devices, wait until all users have finished their immediate operations. For example, all users editing should type the E editor command.

a. Press STOP lever switch.
b. Fill out software trouble report.
c. Press I/O RESET lever switch.
d. Set ADDRESS switches to 140.
e. Press START lever switch.

This will not reenter once-only dialogue. It will not DEASSIGN any devices and will not return core to the system. All Teletypes will be in command mode, and each user must type the START monitor command to restart his job.

3. To reinitialize the monitor to the state it was in just after being loaded:

a. Press STOP lever switch.
b. Fill out software trouble report.
c. Press I/O RESET lever switch.
d. Set ADDRESS switches to 143.
e. Press START lever switch.

Step 10, 11, and 12 of the monitor loading procedure must be repeated and all jobs reloaded.

4. To recover in case the monitor itself has been destroyed:

a. Press STOP lever switch.
b. Fill out software trouble report.
c. Press I/O RESET lever switch.
d. Go back to step 1 of monitor loading procedure.

LIST OF STARTING LOCATIONS IN MONITOR

140 First time - Once-only dialogue, initializes system data storage and I/O devices and moves EXEC DDT's symbol table near to the top of existant memory but leaving the last 200 locations free.

Succeeding Times - initializes I/O devices only.

141 Enters EXEC DDT (3000 octal locations) if the operator has indicated that he wants it in the once-only code. This is used to debug the monitor and is useful for installations making modifications to the monitor for their own special purpose.

142 Enters SYSMAK routine (36 locations) which moves job 1 down on top of the monitor and moves the symbol table to top of memory as set in absolute location 37. This routine is useful to load a program which is to replace the monitor, either a new monitor or a nontime-sharing program.

143 Always examines memories to see which exist, resets system data storage, and kills all jobs.

144 Dismisses highest interrupt in progress and runs null job.

145 Reenters once-only dialogue. This may not be possible since the lowest job may be placed on top of part of the once-only code.

146 Starts up system for first time bypassing the once-only code. It is assumed that SYSMAK and EXEC DDT are not needed. This is useful if the console Teletype is not working.

Monitor Storage Layout

monitor itself
SYSMAK
EXEC DDT
once-only dialogue code

user jobs may overlay SYSMAK, EXEC DDT, and once-only dialogue code

APPENDIX 5

EXAMPLES OF USING A USER'S CONSOLE

Underlined text is typed by the user.

```
IJOB
JOB 3 INITIALIZED

CORE 1

GET SYS PIP1
JOB SETUP

START


MTAO:/W←                                        (rewind mag tape)

MTAO:←PTR:/S

MTAO:/W←

TTY:←MTAO:
00010
00020   THIS IS A DEMONSTRATION FILE.
00030   THESE LINES WERE READ IN FROM THE PAPER TAPE READER.
00040   HERE IS THE LAST LINE OF THE FILE.
00050
00060
↑C                                              ("CTRL" key and "C" pressed simultaneously)
ASSIGN MTAO WXYZ
DEVICE MTAO ASSIGNED

START


MTAO:/W←

TTY:/N←WXYZ:

THIS IS A DEMONSTRATION FILE.
THESE LINES WERE READ IN FROM THE PAPER TAPE READER.
HERE IS THE LAST LINE OF THE FILE.


↑C
CORE 10
8 FREE 1K BLOCKS LEFT, NONE ASSIGNED

KJOB
```

APPENDIX 6

PDP-6 FORTRAN II COMPILER OPERATING INSTRUCTIONS

The PDP-6 FORTRAN II Compiler contains two basic sections: a compiler which generates assembly code from the FORTRAN source statements and an assembler which generates relocatable binary programs. The 22K compiler contains both parts in one program. The 9K compiler, however, prepares an intermediate file for input to the assembler which is a separate program (FOLA).

COMMAND STRING FOR 22K COMPILER

The command string is used to specify the input and output file designations for the compiler. The 22K compiler expects up to two output files and one input file. The general form of the command string is

```
FILE1, FILE2←— FILE3
```

FILE1 will contain the relocatable binary output; FILE2 will contain the listing of the compiled output (source, assembly, binary, errors), and FILE3 is the source or input file. Each file may have one of the following forms:

```
DEVICE:
DEVICE: FILENAME
DEVICE: FILENAME.EXTENSION
```

where DEVICE may be any device mnemonic acceptable to the PDP-6 executive system, FILENAME may be up to six letters and/or digits, and EXTENSION may be up to three letters and/or digits.

The file name extensions REL and LST are assumed for FILE1 and FILE2 unless specified otherwise.

Example:

```
PTP: ,DTA3:LIST←—DTA1:SORC.TXT
```

If FILE1 is not desired, the command string should be of the form:

```
,FILE2←— FILE3
```

If FILE2 is not desired, either of the following command strings is valid:

```
FILE1←—FILE3
FILE1,←— FILE3
```

If neither output file is desired, the valid command strings are:

```
 ←— FILE3
, ←—FILE3
```

SWITCHES FOR THE 22K COMPILER

Switches are letters specifying optional and extra functions to be performed by the compiler. These letters may appear within parentheses or after a forward slash. Only a single letter may follow a slash, while more than one letter may appear within parentheses. The switches are as follows:

K	Skip one file on the device (magnetic tape only).
M	Do not print storage map.
N	Do not list errors on Teletype console if there is a listing file.

APPENDIX 7

LIMITATIONS ON 9K FORTRAN II COMPILER

1. Boolean statements are not allowed.
2. Use of * in IF statements is not allowed.

MULTIPROGRAMMING MANUAL INDEX

For the convenience of the reader, the letters 'A', 'B', and 'C' in the Index are arbitrary references to the top, middle, and bottom sections, respectively, of the pages.

digital

EQUIPMENT CORPORATION

MAYNARD, MASSACHUSETTS

9087 PRINTED IN U.S.A. 3-4/66