TSS/8
TIME-SHARING SYSTEM USER'S GUIDE

For additional copies, order No. DEC-T8-MRFB-D from Program Library, Digital Equipment Corporation, Maynard, Massachusetts 01754. Price $3.00

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Printing September 1968
2nd Printing March 1969
3rd Printing (Rev) February 1970

Your attention is invited to the last two pages of this manual. The Reader's Comments page, when filled in and returned, is beneficial to both you and DEC. All comments received are considered when documenting subsequent manuals, and when assistance is required, a knowledgeable DEC representative will contact you. The Software Information page offers you a means of keeping up to date with DEC's software.

CONTENTS

CONTENTS (Cont)

TABLES

TABLES (Cont)

ILLUSTRATIONS

PREFACE

The TSS/8 User's Guide is a console-oriented manual, written especially for the student, technician, engineer, or scientist. This manual presents a functional overview of the operation of TSS/8 (sometimes called Time Shared-8) from the user's viewpoint, with precise instructions on how to make efficient use of the TSS/8 system.

In particular, the first section is intended to introduce the new user to TSS/8 and provide some insights on how TSS/8 enables a single computer to efficiently service the differing needs of many users simultaneously. In later sections, Monitor commands are explained in detail, and illustrated by actual examples. Full descriptions of a number of System Library Programs are also presented.

The user of this Guide can be certain that the system operates exactly as explained because each operation has been verified on TSS/8, with the actual printout included herein.

Documents referenced (available from DEC's Program Library):

TSS/8 System Manoger's Guide, DEC-T8-MBZA-D
Introduction to Programming, C-18
Paper Tape System User's Guide, DEC-08-NGCC-D
FOCAL-8 Programming Manual, DEC-08-AJAD-D
BASIC-8 Programming Manual, DEC-T8-KJZA-D
Symbolic Editor, Programmer's Reference Manual, DEC-08-ESAB-D
PAL-D Assembler, Programmer's Reference Manual, DEC-D8-ASAB-D
4K FORTRAN, Programmer's Reference Manual, DEC-08-AFCO-D
KT08/I Time-Sharing Option, DEC-8I-H-8NA-D

CHAPTER 1
INTRODUCTION

1.1 GENERAL DESCRIPTION

TSS/8 (Time-Sharing System for the PDP-8/I and -8 Computers) is a general-purpose, time-sharing system offering up to 16 users (24 in certain applications) a comprehensive library of System Programs. These programs provide facilities for editing, assembling, compiling, debugging, loading, saving, calling, and executing user programs on-line. Two conversational, interactive systems, FOCAL-8 and BASIC-8† are also included.

By separating the central processing operations from time-consuming interactions with human users, the computer can, in effect, work on a number of programs simultaneously. Cycling between programs and giving only a fraction of a second at a time to each program or task, the computer can deal with many users seemingly at once. The appearance is created that each user has the computer to himself. The execution of various programs is done without their interfering with each other and without lengthy delays in the response to individual users.

The heart of TSS/8 is a complex of subprograms called the Monitor. Monitor coordinates the operations of the various programs and user consoles, ensuring that the user is in contact at all times with his program. Monitor allocates the time and services of the computer to the various users; it grants a slice of processing (computing) time to each job, and schedules jobs in sequential order to make most efficient use of the system device (disk). Monitor handles user requests for hardware operations (reader, punch, etc.), swaps (moves) programs between memory and disk, and manages the user's private files.

1.2 USER PROGRAMS

When the user is working on a program with TSS/8, his work exists in the computer as though he had his own 4K (4096 word) PDP-8. Several users can run programs at virtually the same time, because TSS/8 Monitor controls the scheduling of execution times. Monitor brings a program into core from the disk, allows it to execute for a short time, and takes note of the state at which execution is stopped. Monitor then brings the next user program into core, and repeats the process. The user is allotted a 4K block of storage that contains his particular

†FOCAL-8 (FOrmula CALculator) is an on-line conversational program developed by DEC. BASIC-8 is a slightly modified version of the algebraic language originally developed at Dartmouth College.

program; this 4K block will be swapped from core onto a 4K area of disk storage when it is necessary for Monitor to bring in another program to run.

After the user's program has been executed, for a period of time it is placed at the end of the queue (line) of user programs waiting to run. If only one program is ready to run, it is allowed to do so without interruption until another program is ready.

If a user wishes to maintain a permanent copy of his program, it is necessary to save a copy within the file area of the disk (an area separate from the swapping area). Later sections of this manual describe the procedures to create and update such files.

1.3 USER FILES

A TSS/8 user is any person logged in on TSS/8. Each user has an account number and password assigned to him by the installation manager or the person responsible for his particular TSS/8; the account number and password allows the user access to the computer. His account number is also used to identify whatever files the user may own within the TSS/8 file system.

The disk (a large external memory device used for storage of programs and data) is divided into logical areas called files. A user can create files and store them in the file storage area of disk. The user can also specify which groups of users may access his files and for what purpose (read, write, or both).

Parts of the disk are used to store System Files; those programs which are accessible to anyone using the computer. A major portion of this manual deals with how to use System Files, generally referred to as System Library Programs.

With the appropriate Monitor commands, the user can create new files and manipulate old files (extend, reduce or delete them). These commands are explained in Chapter 8. Most individual System Library Programs are able to handle user files as input or output with commands issued at the user's console.

1.4 TSS/8 USER CONSOLE

The user's console is a model 33 ASR Teletype® (Figure 1-1). The console keyboard (Figure 1-2) allows the user to communicate with his programs and Monitor. The paper-tape reader and punch are for paper-tape input and output, while the teleprinter provides a typed copy of user input as well as program and Monitor output. The Teletype controls are described here as they apply to the operation of the computer. Off-line operations are explained in Appendix D.

® Teletype is a registered trademark of the Teletype Corporation.

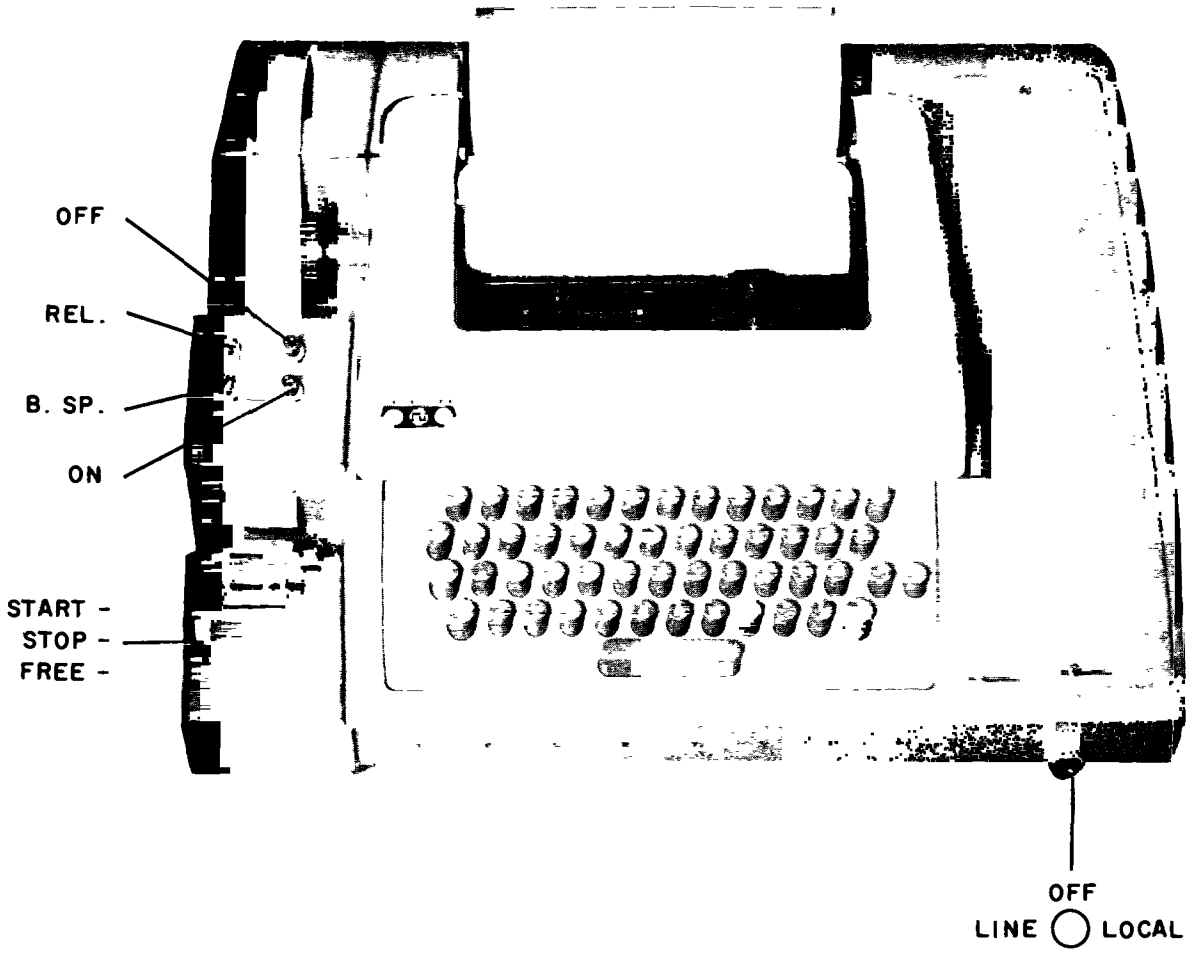


Figure 1-1 User Console

1.4.1 Power Control Knob

The following is a description of settings on the power control knob.

Control	Function
LINE	The Teletype is energized and connected to the computer as an input/output device, under computer control.
OFF	The Teletype is deactivated.
LOCAL	The Teletype is activated for off-line operation.

1.4.2 Printer

The printer provides a typed copy of input and output at a maximum rate of 10 characters per second.

1.4.3 Keyboard

The console keyboard is similar to a typewriter keyboard. However, certain operational functions are shown on the upper part of some of the keytops. These functions are activated by holding down the CTRL key while depressing the desired key. For example, when using the Symbolic Editor, CTRL/FORM causes Editor to enter command mode.

Although the left and right square brackets are not visible on the keyboard keytops, they are shown in Figure 1-2 and are generated by typing SHIFT/K and SHIFT/M, respectively. Also, the ALT MODE key is identified as ESC (ESCape) on some keyboards.

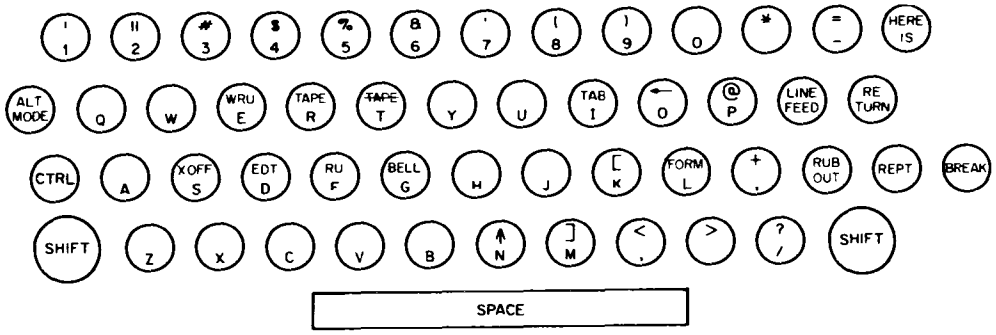


Figure 1-2 Console Keyboard

1.4.4 Paper-Tape Reader and Punch

The paper-tape reader is used to input (to a user program) data punched on eight-channel perforated paper tape at a rate of 10 characters per second (maximum). The reader control settings are shown in Figure 1-1 and described below:

Control	Function
START	Activates the reader; reader sprocket wheel is engaged and operative.
STOP	Deactivates the reader; reader sprocket wheel is engaged but not operative.
FREE	Deactivates the reader; reader sprocket wheel is disengaged.

NOTE

The high-speed reader and punch are described in Appendix E.

1.4.5 Positioning Tape in the Tape Reader

The following procedure describes how to properly position paper tape in the low-speed reader.

a. Raise the tape retainer cover.

b. Set reader control to FREE.

c. Position tape to fit in the carrier and over sprocket wheels.

d. Close the tape retainer cover.

e. Tape should be able to be drawn through low-speed reader in either direction while the control is set to FREE.

1.5 USE OF THIS MANUAL

At this point, the user has a general understanding of time-sharing and how it is done. The following chapters describe how to use the system, and include summaries of the various System Library Programs available and how to use them. Chapter 2 describes the elementary Monitor commands every user will require. Chapter 3, Section 3.1 describes the System Library. Section 3.1 assists the user in learning to gain access to (in computer jargon "to access") files and control execution, and includes a list of more detailed manuals on each of the TSS/8 System Library Programs. A glossary of terms and an Index appears at the end of this manual.

CHAPTER 2
ELEMENTARY MONITOR COMMANDS

TSS/8 offers the user a variety of hardware and software resources. The TSS/8 Monitor controls the allocation and use of these resources. Many of the functions of the Monitor are invisible, and of no concern to the user, for example, the way it allows many users to run programs on a single computer. In other instances the user explicitly tells Monitor what he would like to do and the resources he wishes to utilize. He does so by typing one or more of the commands described in this chapter or Chapter 8 (Advanced Monitor Commands).

The Monitor commands described in the first half of this chapter are those the user needs to log into the system, to utilize the TSS/8 System Library Programs, and to log out of the system. All TSS/8 users must be familiar with these commands. The commands described in sections 2.5, 2.6, and 2.7 are not needed to run TSS/8 System Library Programs such as BASIC and FOCAL, but are frequently useful. The Monitor commands described in Chapter 8 are primarily useful for creating assembly language programs and files.

NOTE

All Monitor commands must be terminated by typing the RETURN key. All words within a Monitor command line are separated by one or more spaces.

2.1 CALLING MONITOR

The user enters commands to System Programs, such as BASIC and FOCAL, in exactly the same way that he enters commands to Monitor (i.e., by typing them in at the keyboard); therefore, the system must have some way of distinguishing between the two cases. It does so by defining two modes of console operation: Monitor mode and program mode. When a user's console is in Monitor mode, all input is interpreted as being commands to Monitor. Otherwise, all input is assumed to be to the user program.

A special character, CTRL/B, (obtained by striking B with the CTRL key held down; and echoed on the Teletype as ↑B) is used to unconditionally place the user's console in Monitor mode. Typing CTRL/B tells the system that the command to follow is to be interpreted as a command to Monitor, regardless of the mode that the Teletype is in. Generally, the command which follows the CTRL/B will be the S command.

↑BS	Return to Monitor mode.
↑B↑BS	Return to Monitor mode from a program which is typing out. (The two CTRL/B's stop the typeout, allowing the S command to be typed.)

It is not necessary to precede each Monitor command with CTRL/B. Once in Monitor mode, a console stays in that mode until a command is typed which starts a user program. To signify that it is in Monitor mode, the system types a dot (.) on the left margin of the console printer paper. This dot indicates that the characters typed in next will be treated as a Monitor command. Thus, the CTRL/B capability is important when a user is running a program and wishes to type a Monitor command. He may, for example, be using one language (or System Program) and want to change to another, as shown below.

```
.R FOCAL

CONGRATULATIONS!!
YOU HAVE SUCCESSFULLY LOADED 'FOCAL,1969' ON A TSS-8/I COMPUTER.


SHALL I RETAIN LOC, EXP, ATN ?:NO

SHALL I RETAIN SINE, COSINE ?:NO

PROCEED.

*
   .
   .
   .
↑BS
.R BASIC

NEW OR OLD--
```

Monitor always responds to ↑BS by typing a dot at the left-hand margin.

2.2 LOGGING IN ON TSS/8

To prevent unauthorized usage and to allow Monitor to maintain a record of system usage, TSS/8 requires that each user identify himself to the system before using it. Before attempting to log in, the user should ensure that the console LINE/OFF/LOCAL knob is turned to the LINE position (see section 1.4.1) before striking the RETURN key. If the console is connected to TSS/8 and is not already in use, Monitor rolls the console paper up two lines and prints a dot at the left margin of the paper.

The dot indicates that TSS/8 is in Monitor mode and that Monitor is waiting for the user to issue a command.

LOGIN	Request access to TSS/8.

The LOGIN command allows the user to access the TSS/8 system.

The user types LOGIN followed by a valid account number and password. Providing the console is free (not already logged in), the command, account number, and password will not be printed on the console paper as the keys are typed. If the command name letters are being printed, stop typing the command; instead, strike the RETURN key, log out using the LOGOUT command (see section 2.3), at this point a successful LOGIN can be accomplished. The LOGIN command is formatted as shown below:

```
.LOGIN 1234 ABCD          (only the dot is printed)
```

where . is printed by Monitor, LOGIN is the command name, 1234 represents the account number, and ABCD represents the password.

NOTE

A command name and each parameter (except the last) is always followed by a space, and the command line is always terminated with the RETURN key.

When a user types something other than a valid LOGIN command on a console, Monitor responds in one of the following ways.

Response	Explanation
. HELLO?	(user typed HELLO)
. LOGIN PLEASE?	(user typed ASSIGN D 3)
. ILLEGAL REQUEST	(user typed LOGIN ABCD ABCD)
.LOGIN 4771 DEMO ALREADY LOGGED IN	(user typed valid LOGIN on an already logged in console)
. UNAUTHORIZED ACCOUNT	(user typed an incorrect account number or password)

In the first example, HELLO is not a command, so it is repeated with a question mark by Monitor. In the second example, ASSIGN D 3 is a valid command but it is not appropriate until after the user logs in; therefore, Monitor asks the user to log in. In the third example, Monitor finds that the LOGIN command is improperly formatted (the first parameter must be from one to four numbers). The console printout tells the user that he has made an ILLEGAL REQUEST. When the console is already logged in and the user types the LOGIN command, the characters typed echo at the console, and Monitor informs the user that the console is occupied with the message ALREADY LOGGED IN? If the user attempts to use an incorrect account number or password, Monitor

replies UNAUTHORIZED ACCOUNT. Thus, Monitor can distinguish an invalid command from a valid command; it can also distinguish whether the valid command is appropriate when issued, whether the command is properly formatted, and whether the account number and password are acceptable. In all the examples above, Monitor ignores the command and prints another dot.

When Monitor finds the LOGIN command properly formatted and the account number and passwords acceptable, it responds by identifying the version of the system being used, the job number it has assigned to the user, the number of the console being used, and the time-of-day in hours, minutes, and seconds. For example:

```
.
TSS/8.19    JOB 03   K01    08:45:21

AND USUALLY THE SYSTEM MANAGER WILL ENTER
HERE A COMMENT OR NOTE TO THE USER CONCERNING
THE SYSTEM

.
```

Monitor then prints another dot and waits for the user to issue the next command. The job number assigned is an internal number by which the system identifies each on-line user.

2.3 LOGGING OUT OF TSS/8

The LOGIN command tells Monitor that the user is ready to begin an on-line session. The LOGOUT command indicates that the user is finished and ready to leave his Teletype.

LOGOUT	Disconnect the user from the system and record the amount of time he has used.

Monitor responds to LOGOUT by typing the amount of computer time used in the session and the total real time of the session. For example:

```
.LOGOUT
RUN TIME 00:00:34   ELAPSED TIME 00:35:41
PLEASE TURN OFF YOUR TTY
```

Computer time used in this example was thirty four seconds, while the elapsed time between LOGIN and LOGOUT was thirty five minutes and forty one seconds.

2.4 SYSTEM LIBRARY PROGRAM CONTROL

Once logged in, the user can call any of the TSS/8 Library Programs described in Chapters 3 through 7. To call such a program, the user types the command R (meaning run) followed by one or more spaces and the pro-

R — Call in and start a TSS/8 System Library Program.

For example:

```
.R BASIC
NEW OR OLD --
```

Monitor fetches the BASIC language processor from the System Library and starts it. BASIC begins its dialogue by asking if the user wishes to work on a new program or retrieve an old one from disk storage. Notice that once BASIC begins, the console is no longer in Monitor mode. Dots are no longer printed at the margin. All input is considered to be commands to BASIC.

If the user types a program name which cannot be found in the System Library, Monitor responds with an error message and returns the console to Monitor mode.

```
.R BASICK
FILE NOT FOUND?
.
```

The exact contents of a TSS/8 System Library may vary from installation to installation.

2.5 COMMUNICATION WITH OTHER USERS

Although TSS/8 gives each system user the impression that he is the only user of the system, it is actually supporting many users at a time. Often, it is useful to communicate with another user, or with the system operator; this is done through use of the TALK command.

TALK — Type out a message on another TSS/8 Teletype.

For example, to tell the system operator (Teletype 0) to turn on the high-speed punch, a user types the following (where the initial dot was typed by Monitor):

```
.TALK Ø PLEASE TURN ON THE HIGH SPEED PUNCH
```

This command causes the following to be typed on console 0:

```
** KØ7**  PLEASE TURN ON THE HIGH SPEED PUNCH
```

where K07 is the number of the physical console which sent the message. Any Teletype can initiate a message to any other Teletype. However, if the destination Teletype is printing at that time, the message will not be sent. The initiating Teletype will, in this case, get the message BUSY as a response.

2.6 SYSTEM STATUS REPORTS

The command SYSTAT initiates a typeout of the full status of TSS/8; how many users are on-line, what they are doing, etc.

SYSTAT	Report system status.

The command SYSTAT is equivalent to typing R SYSTAT. The format of the status report is described in the section on SYSTAT in Chapter 6.

To learn the amount of computer time used since logging in, the user issues the TIME command:

TIME	The elapsed processor time of the user since he logged in is printed.
TIME 0	The time of day is printed.
TIME C1	The amount of processor time used by job C1 since login is printed.

For example:

```
.TIME
00:00:09
.

.TIME 0
09:29:32
.

.TIME 02
00:00:10
.
```

2.7 RESOURCE SHARING

All TSS/8 users, when logged in, have free access to the System Library, the disk storage capability, and the TSS/8 computer. Monitor automatically handles resource requests on a rotating basis. Monitor also maintains a pool of available devices which must be assigned to be used. These are devices, such as the high-speed paper-tape reader, which by their very nature cannot be assigned to several programs simultaneously. Therefore, Monitor grants individual users exclusive access to these devices when needed. Thus, users still share the device, although not simultaneously.

All TSS/8 systems include a high-speed, paper-tape reader in the pool of available devices. Many systems also include a high-speed, paper-tape punch, and one or more DECtapes. These assignable devices are normally used with System Library Programs PIP and COPY to store programs or data on paper tape or DECtape.

When a device is assignable (present on the system) and available (not being used), the ASSIGN command may be used to assign the desired unit or units to the console issuing the command. The valid ASSIGN commands are formatted as shown below:

`ASSIGN R`	Assign the high-speed paper-tape reader.
`ASSIGN P`	Assign the high-speed paper-tape punch.
`ASSIGN D`	Assign a DECtape unit.

where R, P, and D are device designators for reader, punch, and DECtape, respectively. If other devices are assignable, the system manager will inform the user of the appropriate device designators. The following is an example of using an invalid device designator.

```
.ASSIGN X
ILLEGAL REQUEST
```

Monitor ignores the request, responds with the appropriate message, and prints another dot.

When a valid ASSIGN command is issued, Monitor checks for the availability of the device and responds accordingly. For example:

`.ASSIGN R` `R ASSIGNED`	(reader is assignable, available, and assigned)
`.ASSIGN P` `JOB Ø2 HAS P`	(punch is unavailable because job number is using it, and thus not assigned)

When the system contains multiple units of a device, the user simply specifies the device; Monitor assigns an available unit and responds with the unit number. For example:

```
.ASSIGN D
D 2 ASSIGNED
```

If all DECtape units are busy, Monitor prints the message shown below:

```
.ASSIGN D
DEVICE NOT AVAILABLE
```

A specific unit can be requested, leaving a space between the device designator and the device number.

```
.ASSIGN D 4
D 4 ASSIGNED
```
(assignment was accomplished)

NOTE

If the user assigns a device with a nonexistent device number, that device will not be assigned; an error message does not result because that device is not busy. An error message only results when the device is already assigned.

The ASSIGN command can assign only one device at a time. Therefore, when multiple devices are to be assigned, each must be assigned separately. The following will not accomplish the desired assignments, either with or without the illegal commas.

```
.ASSIGN R, D 2, D 1
R   ASSIGNED
```

Monitor accepted the first device designator (and unit number if any) and ignored the rest of the command. If device R is unavailable, Monitor prints the appropriate message. The following completes the desired assignments (assuming available devices).

```
.ASSIGN D 2
D 2 ASSIGNED

.ASSIGN D 1
D 1 ASSIGNED
```

When the user has finished working with an assigned device, the RELEASE command must be used to terminate the assignment and allow other users access to the device. (When a user logs out of TSS/8 any devices he still has assigned to him are automatically released.)

RELEASE — Terminate a previous device assignment and make the device available to other users.

An assigned device is released when the user types the RELEASE command, a space, the device designator (and unit number if required), and the RETURN key as shown below.

```
.RELEASE R
.RELEASE D 3
.
```

In the previous example, the reader and DECtape unit number 3 are released. Monitor prints a dot on the next line if the release is accomplished; otherwise, it prints a message. If, for example, a request is made to release a device which has not been assigned to the issuing console, the following happens:

```
.RELEASE P
ILLEGAL REQUEST
.
```

Monitor printed ILLEGAL REQUEST after it checked and found that the specified device was not assigned to the console issuing the command.

NOTE

All commands must be formatted properly; ILLEGAL REQUEST is printed if the user fails to separate the device designator and unit number with a space.

When multiple device units exist on the system, each must be individually released. For example:

```
.RELEASE D 1
.RELEASE D 2
.RELEASE R
.
```

Monitor does not check when releasing a device as it does when checking to assign an available device. The user could have two device units (for example, two DECtape units) assigned and Monitor would not know which to release; therefore, device numbers are necessary with a RELEASE command. However, where only one unit of a specific device (one reader, one punch, etc.) is on the system, the device designator alone is sufficient. Examples follow.

```
.RELEASE D                    (due to multiple DECtape units)
ILLEGAL REQUEST
.RELEASE R                    (accomplished)
.
.RELEASE D 1                  (D 1 is released)
.
```

2.8 ERROR MESSAGES

An appropriate error message is printed whenever: a Monitor command cannot be performed at the time it was requested, a typing error was made, or the command is illegal (nonexistent). Following each error message, Monitor ignores the request and prints another dot, after which the user can issue another command.

Table 2-1
Monitor Error Messages

Command	Explanation
S1?	The System Interpreter does not understand the command. S1 = command
LOGIN PLEASE?	The user attempted to use a console which is not logged in.
UNAUTHORIZED ACCOUNT	The user attempted to log into the system with an invalid account number or name.
ALREADY LOGGED IN?	The user tried to log in on a console which is already in use.
FULL	The TSS/8 system is full. Another user cannot log in until one of the present on-line users logs out.
TYPE ↑BS FIRST	The user attempted to use a system command which cannot presently be honored due to the status of the user's program. The message may appear even after the user has typed ↑BS, since his program may continue until the I/O in progress at the time of the halt is completed. The user should wait a few seconds and then type his command a second time.
ILLEGAL REQUEST	The user requested an illegal command. This error usually results when some parameter has been given an incorrect value or the request refers to a facility not owned by the user.
BUSY	The user attempted to talk to a console which is currently typing out or is being typed on.

CHAPTER 3

DESCRIPTION OF TSS/8 SYSTEM LIBRARY PROGRAMS AND THE INTERACTIVE LANGUAGES

3.1 INTRODUCTION

The TSS/8 System Library contains a comprehensive set of user programs for a wide range of applications. Language processors, such as BASIC-8, FOCAL, and FORTRAN, allow the user to code and run programs in interactive languages. A complete assembly language system allows programs to be written in PAL-D. Various utility programs perform special functions. The System Library consists of the following programs:

a. BASIC-8 - an easily learned algebraic language originally developed at Dartmouth College.

b. FOCAL - DEC's own conversational language for on-line problem solving.

c. FORTRAN-D - a modified version of FORTRAN.

d. EDIT - a keyboard oriented Symbolic Editor, used to create and modify source programs and data files.

e. PAL-D - a two-pass Symbolic Assembler.

f. LOADER - a Binary Loader used to load assembled programs for execution.

g. ODT - Octal Debugging Technique for testing and modifying assembly language programs.

h. PIP - Peripheral Interchange Program for transferring files between the TSS/8 disk and paper tape.

i. COPY - a utility program used to transfer files between the TSS/8 disk and DECtape.

j. CAT - used to list all the files which a user has stored in his library.

k. SYSTAT - (System Status) a utility program that prints the status of the whole TSS/8 system.

A more detailed description of each of the above System Programs is presented in the following chapters.

3.2 GENERAL CHARACTERISTICS OF SYSTEM LIBRARY PROGRAMS

A fundamental feature of the TSS/8 Monitor is its ability to save programs or other data for each user in his own private library. These individual user libraries are maintained on the system disk. Individual entries in the library are called files, whether they contain programs or data. Within the library itself, there is no distinction between types of files by their contents. Each file is identified with a file name by which it is known, and called into use.

The user does not directly create and update the files in his library. He uses the System Library Programs for this purpose. For example, he can use the SAVE command in BASIC. The SAVE command takes the BASIC program named and saves it as a file in the user's library for future use. Similarly, EDIT can be used to modify an existing file, resulting in the creation of a new file. Therefore, although TSS/8 Monitor provides the actual file storage capability, most file manipulation is done while System Library Programs are being run.

The System Library Programs which operate on these files, must know which file to use, when to create a new file, and what to call it. Each Library Program has its own method of determining whether a user wishes to use an old file or create a new one; this is explained in the sections on individual programs.

Example 1:

```
.R BASIC
NEW OR OLD--OLD
OLD PROGRAM NAME--PRIME
READY
```

Example 2:

```
.R FORT
INPUT:TYPE
OUTPUT:BTYPE
```

For most of his work, the user requires access to only his own library. However, it is often a useful feature to be able to obtain a program from another user's library; allowing a single file to be shared by several users. To access a program from another user's library, the user must tell the system in which individual library the file is stored. The user tells the system by entering the account number of the library's owner. (In the absence of an account number, the user's own library is the assumed source.) To get a file from the System Library, type an asterisk immediately after the file name.

Example 1:

```
.R BASIC
NEW OR OLD--OLD
OLD PRCGRAM NAME--HOSSR*
READY
```

Example 2:

```
.R PALD
 INPUT:NOTPIP 5440
OUTPUT:BIN1
```

NOTE

Most examples in the discussions of individual System Library Programs use file names within the user's own library. The user is free (file protect permitting) to use files from other users' libraries.

Access to another user's files is gained only with his permission. A user may "protect" his files against other users (see Appendix F), i.e., prevent them from gaining access to his files, even though they know his program name and account number. Library Programs never permit a user to write in another user's files. Specifying a file which is protected, or specifying a nonexistent file, is an error that is detected immediately. An error message is typed and the same request made again.

The user places his output in a single file; however, it is often useful to input several files together. (For example, the user may wish to assemble two parts of a PAL-D program together.) To specify more than one input file, separate the file names by commas. No Library Program allows more than three input files. FORTRAN is limited to two; BASIC allows only one.

BASIC is a self-contained programming system, with an editor, compiler, and run-time system. It also has a distinctive disk file format. Files created by BASIC are not compatible with files created by other Library Programs. All other Library Programs depend on each other; therefore, all Library Programs use the same format for their disk files. Consequently, files created by the Editor can be used as input to PAL-D or FORTRAN-D, and numerical files created with the use of the Editor can be read by FORTRAN programs as data files.

Up to this point, only files that exist within the Time-Sharing System, i.e., on the TSS/8 disk, have been described; however, TSS/8 provides two other means of file storage: paper tape and DECtape. The Library Program PIP can be used to transfer files between paper tape and disk. The Library Program COPY allows files to be transferred between disk and DECtape.

3.2.1 Controlling the Execution of System Library Programs

TSS/8 provides the user with two options for stopping the system. CTRL/C (C with the CONTROL key held down) allows the user to stop his BASIC program and return to the beginning of that program without returning to the TSS/8 Monitor. For example, if the user begins to run a BASIC program that has an endless loop, he can type CTRL/C to stop it. BASIC responds to ↑C with READY. All other Library Programs respond in a similar manner.

CTRL/B is used to stop the Library Program most recently called. CTRL/B followed by S and carriage return (RETURN key) unconditionally returns the user to the Monitor level; the user can now call another Library Program. If the system is typing out, two CTRL/Bs and the S (↑B↑BS) are required to stop the system.

RUBOUT is another useful character that deletes the last typed character. Some Library Programs respond bv printing \ or ← while others print the rubbed out character. If the RUBOUT key is typed while entering file names for input or output to a Library Program, RUBOUT deletes the whole line. The request for input or output is then repeated.

3.2.2 Returning to Monitor

The user can stop the execution of a System Library Program at any time by typing CTRL/B followed by S and the RETURN key. The System Library Programs can also initiate a return to the Monitor. When the System Library Programs initate a return, †BS is printed just as though the user had terminated the program. For example, BASIC-8 returns to Monitor when the user types the BYE command:

```
READY

BYE
†BS
.
```

FORTRAN returns to Monitor after completing execution of a program. CAT and SYSTAT return after typing their particular data output. PAL-D returns after completion of an assembly, LOADER at the end of a normal load, and EDIT after completion of an EDIT. FOCAL, BASIC, ODT, PIP, and COPY never return to Monitor; these programs must be terminated by the user.

NOTE

Some System Library Programs return to the Monitor when a fatal error condition is detected.

3.2.3 Additional Manuals

Many of the System Library Programs are documented in their own individual manuals. Those programs for which manuals exist, are:

BASIC-8	DEC-T8-KJZA-D
FOCAL	DEC-08-AJAD-D
EDIT	DEC-08-ESAB-D
FORTRAN-D	DEC-08-AFCO-D
PAL-D	DEC-D8-ASAB-D
ODT	DEC-08-COCO-D

Only a brief description of these programs is given here. Tables of commands are included for reference; however, for detailed instructions on the usage of these programs, the user is referred to the individual manual.

3.3 BASIC-8

BASIC-8 is the TSS/8 version of the time-sharing language BASIC. It allows even the beginning computer user to write and run meaningful programs. BASIC-8 is described at greater length in the BASIC-8 Manual (Order No. DEC-T8-KJZA-D). To call BASIC, type:

```
.R BASIC
```

After the user logs into TSS/8, and calls the BASIC program, BASIC then prints NEW OR OLD--. The user then types the appropriate adjective: NEW (if the user is about to type a new program), or OLD (if the user wants to access a program that was previously filed).

BASIC asks NEW PROGRAM NAME-- (or OLD PROGRAM NAME--) and the user types any combination of six letters or less. If the user is recalling an old program file from memory, he must use exactly the same name as when he originally instructed BASIC to save it.

BASIC prints READY to signal the start of the editing phase; the user then begins to type the new program. If the user types a line consisting of only a line number followed by the RETURN key, that line is deleted. Make certain that each line begins with a line number greater than 0 and less than 2047 and contains no non-digit characters. To enter the line to the computer be sure to strike the RETURN key at the completion of each line.

If, while typing a statement, the user makes a typing error and notices it immediately; he can correct it by striking the RUBOUT key (right-hand side of the keyboard), or the back arrow key (SHIFT/O). Striking either key deletes the character in the preceding space and prints a left arrow (←) for each rubout. The user can then type in the correct character. Striking the RUBOUT key a number of times erases one character from the current line, (spaces are characters) to the left for each RUBOUT typed.

NOTE

BASIC sometimes takes several seconds to accomplish a rubout.

While BASIC is in the editing phase, certain additional commands (which must not have line numbers) are available and are described below:

a. If the user types SAVE followed by one or more spaces, followed by a name, and strikes the RETURN key, the current program is saved for future use under that name.

b. If the user types UNSAVE, followed by a name, and strikes the RETURN key, the program with the name given is deleted from the user's permanent file.

c. If the user types CATALOG, and strikes the RETURN key, a listing of all the program names in his permanent file is typed.

NOTE

Names of temporary files are also shown. These are of the form, BASnn, and can be ignored.

d. If the user types LIST followed by two line numbers separated by a comma, a listing of the particular part of his current program that lies between those line numbers is typed. If the comma and second line number are omitted, only the single line indicated is listed. If no line numbers follow the word LIST (but only the RETURN key), the whole program is listed.

e. If the user types DELETE followed by two line numbers separated by a comma, all lines between and including the two indicated are deleted. If the comma and second line number are omitted, only the single line specified is omitted.

f. When the user is ready to leave the Teletype, he signs off by typing BYE. BYE concludes operations and TTS/8 Monitor deletes any tempory files assigned to the user.

g. After BASIC is called, the user can load a BASIC program from paper tape by placing the tape in the console tape reader and typing TAPE. The tape is read in without printing on the console paper. All RUBOUT characters are ignored.

h. After loading a tape, type KEY to return to normal keyboard mode. (Characters typed while in TAPE mode do not echo on the console paper.)

If information (other than the RETURN key) is to follow the control word, at least one blank must precede the additional information.

3.3.1 Example of a BASIC-8 Program

```
LIST
  10 REM - PROGRAM TO COMPUTE INTEREST ON A LOAN
  20 PRINT "INTEREST IN PERCENT";
  30 INPUT J
  40 LET J=J/100
  50 PRINT "AMOUNT OF LOAN";
  60 INPUT A
  70 PRINT "NUMBER OF YEARS";
  80 INPUT N
  90 PRINT "NUMBER OF PAYMENTS PER YEAR";
 100 INPUT M
 110 LET N=N*M
 120 LET I=J/M
 130 LET B=1+I
 140 LET R=A*I/(1-1/B↑N)
 150 PRINT "MONTHLY PAYMENT ="R
 160 PRINT "TOTAL INTEREST ="R*N-A
 170 END
```

```
READY

RUN

INTEREST IN PERCENT? 8
AMOUNT OF LOAN? 25000
NUMBER OF YEARS? 20
NUMBER OF PAYMENTS PER YEAR? 12
MONTHLY PAYMENT = 209.1103
TOTAL INTEREST = 25186.46


READY
```

3.3.2 Summary of BASIC-8 Statements

Table 3-1 is a summary of BASIC-8 statements.

The following is a list of symbols used in Table 3-1.

v = variable
f = formula
r = relationship
a = arguments
n = line number
s = integer subscript value
d = data, either real or integer

Table 3-1
Summary of BASIC-8 Statements

Statement	Explanation
LET v = f	Assign the value of the formula to the specified variable.
DATA d, ..., d	DATA statements are used to supply one or more numbers to be accessed by READ statements.
READ v, ..., v	READ statements, in turn, assign the next available number in the DATA string to the variables listed.
PRINT a, ..., a	Type the values of the specified arguments, which may be variables, text, or format control characters.
GO TO n	Transfer control to the line number specified and continue execution from that point.
{IF (f r f) THEN n IF (f r f) GO TO n}	If the stated relationship is true, then transfer control to the line number specified; if not, continue in sequence.
FOR v = f_1 TO f_2 STEP f_3	Used for looping repetitively through a series of steps. The FOR statement initializes the variable to the value of formula $_1$. If the increment is positive and the variable $\leq$formula $_2$, the instructions following are executed until the NEXT statement is encountered.

Table 3-1 (Cont)
Summary of BASIC-8 Statements

Statement	Explanation
NEXT v	The NEXT statement increments the variable by the value of formula $_3$ (if omitted, the increment value is +1). The variable is again tested as described above, and this process continues until the loop is repeated the specified number of times. When the variable becomes larger than formula $_2$, control goes to the statement following the NEXT. If the increment (formula $_3$) is negative, then the instructions between the FOR and NEXT statements are executed until the variable becomes less than the value of formula $_2$.
DIM v (s)	Enables the user to enter a table or array with the specified number of elements.
END	Last statement to be executed in the program. This statement must be present.
RANDOMIZE	When placed at the start of a program, causes a different set of random numbers to be generated each time that program, using RND (X), is run.
GOSUB n	Transfers control to the subroutine beginning at the line number indicated.
RETURN	RETURN simplifies the execution of a subroutine by providing an automatic return from the subroutine to the next sequential statement following the appropriate GOSUB (the GOSUB which sent control to the subroutine).
INPUT v, ..., v	Causes typeout of a ? to the user and waits for user to respond by typing the value of the variable(s).
STOP	Equivalent to GO TO [line number of END statement].
REM	Permits typing of remarks within the program.
RESTORE	Sets pointer back to beginning of string of DATA values.

3.3.3 Functions

BASIC performs several mathematical calculations for the programmer eliminating the need for tables of trigonometric functions, square roots and logarithms. These functions have a three-letter call name, (the argument (X) can be a number, variable or formula) and are written as follows:

Functions	Meaning
SIN (x)	sine of x in radians
COS (x)	cosine of x in radians
TAN (x)	tangent of x in radians
ATN (x)	arctangent of x in radians
EXP (x)	e^x where e=2.712818
LOG (x)	natural logarithm of x, $\log_e x$
ABS (x)	absolute value of x, $\|x\|$
SQR (x)	square root of x, $\sqrt{x}$

3.3.4 Complex Functions

a. SGN (x) - the SGN or sign function returns the value +1 if x is a positive number, 0 if x is 0, and -1 if x is negative. For example: SGN (3.42) =1, SGN (-42) =-1.

b. INT (x) - the INT or integer function returns the value of the greatest integer not greater than x. For example: INT (34.67) =34, INT (-34.42) =-35.

c. RND (x) - the RND or random number function produces a random number between 0 and 1. The numbers are reproducible in the same order for later checking of a program. The value of x is ignored. If a truly random number generator (causing different random numbers every time the program runs) is desired, place the RANDOMIZE statement at the beginning of the program, before the initial use of RND (x).

d. User defined functions - The user may create his own functions by using the DEF statement. The statement defining the function must appear before any reference to that function, its form is: DEF FNA (x) = (some formula) where the function name must be a three-letter sequence beginning with F.

3.3.5 Summary of Edit and Control Commands

Several commands for editing BASIC programs and controlling their execution enable the user to: delete lines, list his program, save programs on a file-structured storage device (disk), delete or replace old programs on the storage device with new programs, call in programs from the storage device, etc. These commands are summarized below.

Table 3-2
Summary of BASIC Edit and Control Commands

Command	Action
BYE	Exit to TSS/8 Monitor to conclude operations
DELETE n	Delete line number n (or simply type the line number and RETURN key).
DELETE m, n	Delete line numbers m through n.
LIST	List program.
LIST n	List line number n.
LIST m, n	List program from line number m through n.
NEW	BASIC will ask for a new program name.
OLD	BASIC will ask for the old program name and will replace current contents of user core with the program of the given name from the storage device.
RUN	Compile and run program currently in core.
SAVE name	Save the BASIC program currently being worked on under the name given. †
UNSAVE name	Delete the named program from the storage device.
↑C (CTRL/C)	To stop a running program, type CTRL/C (C with the CONTROL key held down). BASIC will return to editing mode and type READY.

†SAVE commands overwrite an existing file of the same name.

3.3.6 Error Messages

There are four types of BASIC error messages. The messages and their various interpretations are shown below.

3.3.6.1 Editing Phase Diagnostics - Retype the line to correct it.

Error Message	Explanation
SYSTEM I-O ERROR	BASIC was unable to perform the desired disk I/O.
//ERROR Ø1	The user did not type in OLD or NEW when the information was requested.
//ERROR Ø2	The new or old name is not a valid name.
//ERROR Ø3	The new name given is currently an active program.
//ERROR Ø4	The old program name requested cannot be found.
//ERROR 11	The SAVE or UNSAVE name given is not a valid name.
WHAT?	The editor cannot understand the command given.
//ERROR 2Ø	Invalid line number format or outside the range 0 < line number < 2047.

3.3.6.2 Compilation and Execution Diagnostics - Most messages are followed by the notation ON LINE nnnn, where nnnn is the line number on which the error was detected. (BASIC prints READY and the user is back in the editing phase.)

Error Message	Explanation
PROGRAM TOO LARGE	The program is too large to be executed. Make it smaller.
MISSING END STATEMENT	All programs must have an END statement.
DATA POOL OVERFLOW	The user used too many constants and/or variables in the program.
ILLEGAL STATEMENT	A statement was used which is not one of the legitimate BASIC statements.
ILLEGAL LINE FORMAT	The structure of the statement does not agree with the BASIC syntax.
NOT CONSTANT IN DATA	The user attempted to use something other than a constant in a DATA statement.
DEF STATEMENT MISSING	A function needing a DEF statement exists in the program.
FOR WITHOUT NEXT	There is an unmatched FOR statement in the program.
NEXT WITHOUT FOR	The NEXT statement indicated has no preceding FOR statement.
ILLEGAL CHARACTER	The user attempted to use an illegal character in the statement being processed.
ILLEGAL CONSTANT	The format of a constant in the statement being processed is not valid.
INVALID NAME	A name is being used which does not agree with the BASIC requirements.
INVALID LINE NUMBER	The format of the line number, being used in a GO TO or IF statement, is not acceptable.
ARRAY USED BEFORE DEFINED	The user attempted to use an array prior to its appearance in a DIM statement.
EXPRESSION SYNTAX	The expression being processed does not agree with the BASIC rules (this will probably be due to unmatched parentheses).
STACK OVERFLOW	The user programmed a situation in which either DO statements, subroutines, or functions are nested too deeply; or the user has a function which calls itself.
OUT OF DATA	An attempt was made to READ more data than was supplied by the user.
ILLEGAL INPUT FORMAT	The form of a constant, which the user is attempting to INPUT, is not valid.
DIMENSION TOO LARGE	Too large an array to fit in the core available.
UNDEFINED LINE NUMBER	The line number appearing in a GO TO or an IF-THEN statement does not appear in the program.

3.3.6.3 Non-Fatal Execution Errors - Non-fatal errors are for notification purposes and indicate that the user performed a computational range error. The errors all cause the message XX IN nnnn to be printed, where nnnn is the line number and XX is as described below:

Error Code	Explanation
/Ø	ZERO DIVIDE - An attempt was made to divide a number by zero. The largest possible number is used for the result.
OV	OVERFLOW - The result of a calculation was too large for the computer to handle. The largest possible number is used for the result.
UN	UNDERFLOW - The result of a calculation was too small for the computer to handle. Zero is used for the result.
LN	An attempt was made to compute the logarithm of zero or a negative number. Zero is used for the result.
SQ	An attempt was made to compute the square root of a negative number. The square root of the absolute value is used for the result.
PW	An attempt was made to raise a negative number to a fractional power. The absolute value of that number raised to the fractional power is used.

3.3.6.4 System Error - If a failure occurs in the disk I/O portion of the BASIC system, the message SYSTEM I/O ERROR is printed and control returns to the editing phase.

3.3.7 Implementation Notes

The TSS/8 BASIC language is compatible with Dartmouth BASIC except as noted below:

a. There are no matrix operations.

b. There are no character string instructions.

c. The ON statement has not been implemented.

d. The TAB function is not available in PRINT statements.

e. BASIC-8 has no features which allow reading or writing data on the disk. (Although programs may be saved on the disk for future use.)

f. All array (subscripted) variables must appear in a DIM statement.

g. The function INT (X) gives the greatest integer in X. Therefore, INT (-2.3) gives the value -3; INT (2.3) gives the value +2

h. Negative numbers can not be raised to integer powers. The absolute value is used and an error message printed. The reason for this is the unary minus has a lower priority during execution than exponentiation. For example, $-2^3 = +8$ with an error message given.

i. User defined functions are restricted to one line.

j. Maximum size of a BASIC-8 program is about 300 lines, depending on the number of variables, number and size of arrays, and number of nested subroutines and FOR-NEXT loops.

3.4 FOCAL

FOCAL (Formulating On-line Calculations in Algebraic Language) is an on-line, conversational, service program for the PDP-8 family of computers, designed to help scientists, engineers, and students solve numerical problems. The language consists of short imperative English statements which are relatively easy to learn. It is used for simulating mathematical models, for curve plotting, for handling sets of simultaneous equations, and many other kinds of problems.

For a detailed introduction to the FOCAL programming language, consult the FOCAL Programming Manual (Order No. DEC-08-AJAD-D).

To call FOCAL, type:

```
.R FOCAL
```

FOCAL enters its initial dialogue, and asks if its extended functions are to be retained. The extended functions are exponential, sine, cosine, arctangent, and logarithm. If the FOCAL program to be run uses any of these functions, the user responds YES. If not, the user responds NO to free more space for the user program. Without the extended functions, there is room for approximately 1800 characters of program. If the extended functions are retained, there is room for approximately 1100 characters.

3.4.1 FOCAL Command and Operation Summary

Table 3-3
FOCAL Command and Operation Summary

Command	Abbreviation	Example of Form	Explanation
ASK	A	ASK X, Y, Z	FOCAL types a colon for each variable; the user types a value to define each variable.
COMMENT	C	COMMENT	If a line begins with the letter C, the remainder of the line will be ignored.
CONTINUE	C	C	Dummy lines.

Table 3-3 (Cont)
FOCAL Command and Operation Summary

Command	Abbreviation	Example of Form	Explanation
DO	D	DO 4.1	Execute line 4.1; return to command following DO command.
		DO 4.0 DO ALL	Execute all group 4 lines; return to command following DO command, or when a RETURN is encountered.
ERASE	E	ERASE	Erases the symbol table.
		ERASE 2.0	Erases all group 2 lines.
		ERASE 2.1	Deletes line 2.1.
		ERASE ALL	Deletes all user input.
FOR	F	FOR i=x,y,z;(commands)	Where the command following is executed at each new value. x=initial value of i y=value added to i until i is greater than z.
GO	G	GO	Starts indirect program at lowest numbered line number.
GO?	G?	GO?	Starts at lowest numbered line number and traces entire indirect program until another ? is encountered, until an error is encountered, or until completion of program.
GOTO	G	GOTO 3.4	Starts indirect program (transfers control to line 3.4). Must have argument.

IF	I	IF (X)Ln, Ln, Ln
		IF (X)Ln, Ln; (commands)
		IF (X)Ln; (commands)
			Where X is a defined identifier, a value, or an expression, followed by three line numbers.
			If X is less than zero, control is transferred to the first line number.
			If X is equal to zero, control is to the second line number.
			If X is greater than zero, control is to the third line number.

MODIFY	M	MODIFY 1.15	Enables editing of any character on line 1.15 (see below).
QUIT	Q	QUIT	Returns control to the user.
RETURN	R	RETURN	Terminates DO subroutines, returning to the original sequence.

Table 3-3 (Cont)
FOCAL Command and Operation Summary

Command	Abbreviation	Example of Form	Explanation
SET	S	SET A=5/B*C;	Defines identifiers in the symbol table.
TYPE	T	TYPE A+B-C;	Evaluates expression and types out = and result in current output format.
		TYPE A-B, C/E;	Computes and types each expression separated by commas.
		TYPE "TEXT STRING"	Types text. May be followed by ! to generate a carriage return and line feed, or # to generate a carriage return.
WRITE	W	WRITE WRITE ALL	FOCAL types out the entire indirect program.
		WRITE 1.0	FOCAL types out all group 1 lines.
		WRITE 1.1	FOCAL types out line 1.1.

3.4.2 FOCAL Operations

The following is a description of symbols used in FOCAL operation.

	Symbol	Explanation
To set output format,	TYPE %x.y	Where x is the total number of digits, and y is the number of digits to the right of the decimal point.
	TYPE %6.3, 123.456	FOCAL types: =+123.456
	TYPE %	Resets output format to floating point.
To type symbol table,	TYPE $	Other statements may not follow on this line.

After a MODIFY command, the user types a search character, and FOCAL types out the contents of that line until the search character is typed. The user may then perform any of the following operations.

a. Type in new characters. FOCAL will add these to the line at the point of insertion.

b. Type a CTRL/L. FOCAL will proceed to the next occurrence of the search character.

c. Type a CTRL/BELL. After this, the user may change the search character.

d. Type RUBOUT. This deletes characters to the left, one character for each time the user strikes the RUBOUT key.

e. Type ←. Deletes the line over to the left margin, but not the line number.

f. Type RETURN. Terminates the line, deleting characters over to the right margin.

g. Type LINE FEED. Saves the remainder of the line from the point at which LINE FEED is typed over to the right margin.

3.4.3 Mathematical Functions

A listing of mathematical functions follows:

Function	Symbol	Interpretation
Square Root	FSQT(x)	where x is a positive number or expression greater than zero.
Absolute Value	FABS(x)	FOCAL ignores the sign of x.
Sign Part	FSGN(x)	FOCAL evaluates the sign part only, with 1.0000 as integer.
Integer Part	FITR(x)	FOCAL operates on the integer part of x, ignoring any fractional part.
Random Number Generator	FRAN(x)	FOCAL generates a random number. The value of x is ignored.
†Exponential Function (e^x)	FEXP(x)	FOCAL generates e to the power x. (2.71828^x)
†Sine	FSIN(x)	FOCAL generates the sine of x in radians.
†Cosine	FCOS(x)	FOCAL generates the cosine of x in radians.
†Arc Tangent	FATN(x)	FOCAL generates the arc tangent of x in radians.
†Logarithm	FLOG(x)	FOCAL generates $\log_e$ (x).

3.4.4 Control Characters

Control characters and their explanation follows:

%	Output format delimiter	
!	Carriage return and line feed	
#	Carriage return	
$	Type symbol table contents	
( )	Parentheses	(mathematics)
[ ]	Square brackets	
< >	Angle brackets	
" "	Quotation marks	(text string)
? ?	Question marks	(trace feature)
*	Asterisk	(high-speed reader input)

†These are known as extended functions.

SPACE key (names)
RETURN key (lines)
ALT MODE key (with ASK statement) } (nonprinting)
Comma (expressions)
Semicolon (commands and statements)

3.4.5 Reading FOCAL Paper Tapes

To ensure that FOCAL paper tapes are read without error, they should be read silently. To do this, type ↑B (CTRL/B) followed by UNDUPLEX just prior to reading the tape in. This Monitor command suppresses the printing of the program as it is read. As each line is read, a line feed and FOCAL's asterisk are typed, indicating that the line is properly stored. After the tape has been completely read, type ↑B DUPLEX to restore FOCAL to its normal mode. An example is shown below:

```
*ERASE ALL
*↑BUNDUPLEX


*
 *
  *
   *
    *
     *
      *
       *↑BDUPLEX


*WRITE ALL
C-FOCAL,1969

01.05 C PROGRAM TO CALCULATE THE HYPOTENUSE OF A
01.06 C RIGHT TRIANGLE GIVEN THE TWO SIDES
01.10 ASK "SIDES OF TRIANGLE ARE" A,B
01.20 SET C=FSQT(A↑2+B↑2)
01.30 TYPE "HYPOTENUSE IS" C,!
01.40 GOTO 1.1
*
```

3.4.6 FOCAL Error Messages

Code	Explanation
?00.00	Manual start given from console.
?00.00	Interrupt from keyboard via CTRL/C.
?01.40	Illegal step or line number used.
?01.78	Group number is too large.
?01.96	Double periods found in a line number.
?01.;5	Line number is too large.
?01.;4	Group zero is an illegal line number.
?02.32	Nonexistent group referenced by DO.
?02.52	Nonexistent line referenced by DO.

Code	Explanation
?02.79	Storage was filled by push-down list.
?03.05	Nonexistent line used after GOTO or IF.
?03.28	Illegal command used.
?04.39	Left of = in error in FOR or SET.
?04.52	Excess right terminators encountered.
?04.60	Illegal terminator in FOR command.
?04.:3	Missing argument in display command.
?05.48	Bad argument to MODIFY.
?06.06	Illegal use of function or number.
?06.54	Storage is filled by variables.
?07.22	Operator missing in expression or double E.
?07.38	No operator used before parenthesis.
?07.:9	No argument given after function call.
?07.;6	Illegal function name or double operators.
?08.47	Parentheses do not match.
?09.11	Bad argument in ERASE.
?10.:5	Storage was filled by text.
?11.35	Input buffer has overflowed.
?20.34	Logarithm of zero requested.
?23.36	Literal number is too large.
?26.99	Exponent is too large or negative.
?28.73	Division by zero requested.
?30.05	Imaginary square roots required.
?31.<7	Illegal character, unavailable command, or unavailable function used.

CHAPTER 4
FORTRAN-D COMPILER

4.1 INTRODUCTION TO TSS/8 FORTRAN

FORTRAN-D compiles and runs programs written in the PDP-8 version of FORTRAN. Programs (usually created and stored with the Symbolic Editor) are compiled in a single pass and executed (automatically) immediately following compilation. The PDP-8 FORTRAN Manual (Order No. DEC-08-AFCO-D) provides exact instructions for writing programs in FORTRAN.

4.2 CALLING AND USING FORTRAN-D

To use FORTRAN-D, type:

```
. R FORT
```

FORTRAN requests the name of the input file, i.e., the file containing the FORTRAN program to be compiled and run. The user responds with the file name, then strikes the RETURN key. FORTRAN then requests the name of an output file in which to store the compiled version of the program. For normal usage, just the RETURN key need by typed. FORTRAN places the compiled code in a file of its own, then proceeds to run the program.

If a file name is entered for output, FORTRAN creates a permanent file in which the compiled binary program is saved. It is then possible to rerun this program without recompiling it. To run an already compiled program, call the FORTRAN operating system directly by typing:

```
.R FOSL
```

FOSL requests the name of an input file. Enter the name of the file containing the compiled binary.

Examples:

```
.R FORT

INPUT:MTRIX
OUTPUT:
```

FORTRAN compiles and executes the program MTRIX but does not save the compiled binary.

```
.R FORT

INPUT:MTRIX
OUTPUT:BMTRIX
```

FORTRAN compiles and executes the program MTRIX and then leaves the compiled binary in the file named BMTRIX.

```
.R FOSL

INPUT:BMTRIX
```

The FORTRAN binary program BMTRIX is executed without first being compiled.

NOTE

All FORTRAN programs return to the Monitor when they have completed execution.

4.3 FORTRAN I/O

Differing versions of PDP-8 FORTRAN offer slightly different features. TSS/8 FORTRAN-D differs in the way it is called into use (described above), and in its more powerful I/O capability (described below). FORTRAN-D allows three data formats:

I	Integer format
E	Exponential format
A	Alpha format, the ASCII value of a character is stored as an integer value.

The standard device for READ and WRITE statements is the Teletype, which is assigned device code 1. Because the Teletype is so frequently used, FORTRAN-D includes two special input/output instructions, ACCEPT and TYPE. These instructions imply use of the Teletype; therefore, the device code need not be specified. ACCEPT is especially convenient if data is to be entered at the keyboard because this instruction automatically supplies a line feed when the RETURN key is typed. Also, the user can correct an erroneously typed value by striking the RUBOUT key.

A FORTRAN-D program can also utilize the high-speed reader and punch for I/O. These devices are assigned code 2. Because the high-speed reader and punch are shared by all TSS/8 users, it is necessary to assign them if they are to be used. Assign the appropriate devices and mount tapes in the reader before running FORTRAN-D. When running several FORTRAN-D programs, reassign the devices before each run.

TSS/8 FORTRAN also allows programs to read and write data files on the disk. These data files are completely separate from the program files. Data files are read and written by standard READ and WRITE statements within the FORTRAN-D program. The device code for the disk is 3. Because programs using the disk are treated differently by FORT (the FORTRAN-D compiler), it is necessary to identify programs which use the disk. These programs are identified by a DEFINE DISK statement as the first statement in any such FORTRAN-D program including a READ or WRITE statement with device code 3.

Just as FORT itself must ask for the name of its input and output files, so must a FORTRAN program ask for the names of its disk files. FORTRAN-D programs do this by typing INPUT: and OUTPUT: a second time. The user responds by typing the name of the files to be read or written by the program. FORTRAN-D asks for both input and output for all programs which include a DEFINE DISK statement. If only input (or output) is to be used, the user responds to the other by striking the RETURN key.

The following program is an example of a FORTRAN-D program which utilizes the disk for data storage. The program reads an already existing file (DATA1) of 10 values from the disk and writes their square roots into a new file (DATA2).

4.4 EXAMPLES OF FORTRAN PROGRAMS

Example 1:

```
C       FORTRAN PROGRAM TO READ 10 VALUES FROM A DISK FILE,
C       COMPUTE THEIR SQUARES, AND WRITE THE SQUARES OUT TO
C       A SECCND DISK FILE.
C
        DEFINE DISK
        DIMENSION A(10)
C
        DO 20 I=1,10
        READ 3,10,A(I)
10      FORMAT (E)
20      CONTINUE
C
        DO 30 I=1,10
        A(I)=A(I)**2
30      CONTINUE
C
        DO 40 I=1,10
        WRITE 3,10,A(I)
40      CONTINUE
C
        STOP
        END

.R FORT

INPUT:SQUARE
OUTPUT:
```

```
INPUT:DATA1
OUTPUT:DATA2


↑BS
.
```

Example 2:

```
.R EDIT

 INPUT:MTRIX*
OUTPUT:
R

L


C       MATRIX MULTIPLIER
        DIMENSION A(36), B(36), C(36)
        TYPE 200
        TYPE 201
        READ 1,1,I
1       FORMAT (I)
        TYPE 202
        DO 10 M=1,I
        DO 10 N=1,I
        INDX=M+I*(N-1)
        READ 1,2,A(INDX)
2       FORMAT (E)
10      CONTINUE
        TYPE 15
15      FORMAT (/)
        TYPE 202
        DO 20 M=1,I
        DO 20 N=1,I
        INDX=M+I*(N-1)
        READ 1,2,B(INDX)
        C(INDX)=0
20      CONTINUE
        DO 30 M=1,I
        DO 30 N=1,I
        DO 30 K=1,I
        IC=N+I*(M-1)
        IA=M+I*(K-1)
        IB=K+I*(N-1)
        C(IC)=C(IC)+A(IA)*B(IB)
30      CONTINUE
        TYPE 15
        DO 40 M=1,I
        TYPE 21
        DO 40 N=1,I
        INDX=N+I*(M-1)
        TYPE 2, C(INDX)
40      CONTINUE
21      FORMAT (/)
        TYPE 15
```

```
200     FORMAT (/,"MATRIX MULTIPLIER",/)
201     FORMAT ("DIMENSION IS:")
202     FORMAT (/,"ENTER MATRIX",/)
        END


↑BS
•

.R FORT

         INPUT:MTRIX*
        OUTPUT:

        MATRIX MULTIPLIER
        DIMENSION IS:3
        ENTER MATRIX
        12 23 34
        11 21 13
        10 20 30

        ENTER MATRIX
        1.2 1.2 3.1
        .01 121 12.
        111 1.2 343

        0.378863E+4     0.283820E+4     0.119751E+5
        0.145641E+4     0.256979E+4     0.474510E+4
        0.334219E+4     0.246800E+4     0.105610E+5

        ↑BS
        •
```

4.5 SUMMARY OF FORTRAN-D STATEMENTS

Table 4-1
Summary of FORTRAN-D Statements

Statement and Form	Explanation
Arithmetic Statements	
v = e	v is a variable (possibly subscripted); e is an expression.
Control Statements	
GO TO n	n is a statement number.
GO TO $(n_1,n_2,\ldots,n_n),i$	$n_1,\ldots,n_n$ are statement numbers; i is a nonsubscripted integer variable.
IF (e) n_1,n_2,n_3	e is an expression; n_1,n_2,n_3 are statement numbers.
DO n i=k_1,k_2,k_3	n is the statement number of a CONTINUE; is is an integer variable; k_1,k_2,k_3 are integers or nonsubscripted interger variables.
CONTINUE	Proceed
PAUSE	Temporarily suspend execution.

Table 4-1 (Cont)
Summary of FORTRAN-D Statements

Statement and Form	Explanation
STOP	Terminate execution.
END	Terminate compilation; last statement in program.
Specification statements	
DIMENSION $v_1(n_1),v_2(n_2),\ldots,v_n(n_n)$	$v_1,\ldots,v_n$ are variable names; $n_1,\ldots,n_n$ are integers.
DEFINE device	Device is DISK or TAPE, system I/O device.
FORMAT $(s_1,s_2,\ldots,s_n)$	s is a data field specification.
COMMENT	Designated by C as first character on line.
Input/Output Statements	
ACCEPT f,list	f is a FORMAT statement number; list is a list of variables.
TYPE f,list	f is a FORMAT statement number; list is a list of variables.
READ u,f,list	u is an integer, representing device from which data is to be read.
	f is a FORMAT statement number; list is a list of variables.
WRITE u,f,list	u is an integer, representing device onto which data will be written.
	f is a FORMAT statement number; list is a list of variables.

4.6 FORTRAN-D COMPILER SYSTEMS DIAGNOSTICS

Table 4-2
FORTRAN-D Compiler Systems Diagnostics

Error Code	Explanation
0240	System file error. One of the FORTRAN components cannot be found or the disk is full, preventing FORTRAN from proceeding. Try recalling FORT.
3100	Illegal operator on compiler stack †
3417	Pre-precedence error †
6145	Could not find FOSL on system device; if the error occurs, it may be necessary to reload FORT and FOSL.
6223	Error while loading .FT.

†Error may be due to a compiler error or a machine malfunction.

Table 4-2 (Cont)
FORTRAN-D Compiler Systems Diagnostics

Error Code	Explanation
6226	Same as above
6257	Same as above
6724	No END statement on source device
6746	Same as above
7114	Same as above

†Error may be due to a compiler error or a machine malfunction.

4.7 FORTRAN-D COMPILER COMPILATION DIAGNOSTICS

Table 4-3
FORTRAN-D Compiler Compilation Diagnostics

Error Code	Explanation
00	Mixed mode arithmetic expression
01	Missing variable or constant in arithmetic expression
03	Comma was found in an arithmetic expression
04	Too many operators in this expression
05	Function argument is in fixed-point mode
06	Floating-point variable used as a subscript
07	Too many variable names in this program
10	Program too large, core storage exceeded
11	Unbalanced right and left parentheses
12	Illegal character found in this statement
13	Compiler could not identify this statement
14	More than one statement with same statement number
15	Subscripted variable did not appear in a DIMENSION statement
16	Statement too long to process
17	Floating-point operand should have been fixed-point
20	Undefined statement number
21	Too many numbered statements in this program
22	Too many parentheses in this statement
23	Too many statements have been referenced before they appear in the program
25	DEFINE statement was preceded by some executable statement
26	Statement does not begin with a space, tab, C, or number

4.8 FORTRAN-D OPERATING SYSTEM DIAGNOSTICS

Table 4-4
FORTRAN-D Operating System Diagnostics

Error Code	Explanation
01	Checksum error on FORTRAN binary input
02	Illegal origin or data address on FORTRAN binary input
04	Disk input-output error †
05	High-speed reader error
06	Illegal FORTRAN binary input device
11	Attempt to divide by zero
12	Floating-point input data conversion error
13	Illegal op code
14	Disk input-output error †
15	Non-FORMAT statement used as a FORMAT
16	Illegal FORMAT specification
17	Floating-point number larger than 2047
20	Square root of a negative number
21	Exponential negative number
22	Logarithm of a number less than or equal to zero
40	Illegal device code used in READ or WRITE statement
41	System device full, could not complete a WRITE statement
76	Stack underflow error ††
77	Stack overflow error ††

†May be caused by machine malfunction or operating system error.

††May be caused by source program or loading error; to correct, do the following in descending order.

a. Use Diagnose to determine where the error occurred.

b. Recompile the source program.

c. Examine source program (in particular the arithmetic statements and subscripted variables).

CHAPTER 5
PAL-D ASSEMBLER

5.1 INTRODUCTION TO PAL-D

The TSS/8 Assembly System is composed of the PAL-D Symbolic Assembler, LOADER, and ODT. The PAL-D Assembler is used to translate the user's source program into an object program (binary or machine code). LOADER is used to transfer the user's object program from the disk into core for debugging or execution. ODT (Octal Debugging Technique) is used to dynamically debug the object program which has been loaded into core using LOADER.

The PAL-D Assembler is fully documented in the PDP-8 PAL-D Assembler, Programmer's Reference Manual (Order No. DEC-D8-ASAB-D), and its operation under TSS/8 is covered in an appendix in that document. PAL-D source programs are usually written on-line using the EDIT program, which stores them in disk files. The Assembler checks for assembly language syntax errors and for undefined user symbols but does not check for logic errors. To call the PAL-D Assembler, type:

```
.R PALD
```

PAL-D responds by requesting INPUT: Type and enter the name of the source program or programs to be assembled. A maximum of three files can be assembled together. PAL-D then requests OUTPUT: Type in the name of the new file in which PAL-D will store the assembled program in executable binary form. PAL-D then requests OPTION: For a normal assembly, strike the RETURN key. If an assembly listing is not desired, respond to OPTION: with N.

PAL-D then proceeds to assemble the program: any errors in the program are indicated; the program symbol table is printed; and finally, an assembly listing of the source program is printed. When the listing is completed and the assembly finished, control is returned to TSS/8 Monitor.

5.2 TSS/8 PAL-D

Because of the necessary hardware changes made for the Time-Shared System, PAL-D has been revised in the following ways:

a. PAL-D, under TSS/8, allows 245 (decimal) user symbols in addition to the permanent symbols (listed in section 5.4). All symbols reside in locations 5200 through 7777. The permanent symbol table has been revised to include all instructions peculiar to the Time-Sharing System.

b. A CTRL/C (↑C) from the Teletype terminates the assembly, and halts PAL-D, sending the user back to the TSS/8 Monitor.

5.3 EXAMPLE OF A PAL-D PROGRAM

```
.R PALD

 INPUT:TYPE2
OUTPUT:BIN2
OPTION:

COUNT   0415
CRLF    0417
LOOP    0406
OUT     0425
REG     0416
START   0400

                /PROGRAM TO TYPE OUT "0123456789"
                *0400
0400    7200    START,  CLA
0401    4217            JMS CRLF
0402    1377            TAD (-12)
0403    3215            DCA COUNT
0404    1376            TAD (260)       /ASCII CODE FOR ZERO (0)
0405    3216            DCA REG
0406    1216    LOOP,   TAD REG
0407    4225            JMS OUT
0410    2216            ISZ REG
0411    2215            ISZ COUNT
0412    5206            JMP LOOP
0413    4217            JMS CRLF
0414    7402            HLT
0415    0000    COUNT,  0
0416    0000    REG,    0

0417    0000    CRLF,   0
0420    1375            TAD (215)       /ASCII FOR CARRIAGE RETURN
0421    4225            JMS OUT
0422    1374            TAD (212)       /LINE FEED
0423    4225            JMS OUT
0424    5617            JMP I CRLF

0425    0000    OUT,    0
0426    6046            TLS
0427    6041            TSF
0430    5227            JMP .-1
0431    7200            CLA
0432    5625            JMP I OUT
0574    0212
0575    0215
0576    0260
0577    7766
↑BS
.
```

5.4 SYMBOL LIST FOR TSS/8

Table 5-1
Symbol List for TSS/8

Mnemonic	Code	Operation	Event Time
		MEMORY REFERENCE INSTRUCTIONS	
AND	0000	Logical AND	
TAD	1000	Twos complement add	
ISZ	2000	Increment & skip if zero	
DCA	3000	Deposit & clear AC	
JMS	4000	Jump to subroutine	
JMP	5000	Jump	
		GROUP 1 OPERATE MICROINSTRUCTIONS	
NOP	7000	No operation	1
IAC	7001	Increment AC	3
RAL	7004	Rotate AC & link left one	3
RTL	7006	Rotate AC & link left two	3
RAR	7010	Rotate AC & link right one	3
RTR	7012	Rotate AC & link right two	3
CML	7020	Complement link	2
CMA	7040	Complement AC	2
CLL	7100	Clear link	1
CLA	7200	Clear AC	1
		GROUP 2 OPERATE MICROINSTRUCTIONS	
HLT	7402	Halts the computer	4
OST	7404	Inclusive OR switch register with AC	3
SKP	7410	Skip unconditionally	1
SNL	7420	Skip on nonzero link	1
SZL	7430	Skip on zero link	1
SZA	7440	Skip on zero AC	1
SNA	7450	Skip on nonzero AC	1
SMA	7500	Skip on minus AC	1
SPA	7510	Skip on plus AC (zero is positive)	1
		COMBINED OPERATE MICROINSTRUCTIONS	
CIA	7041	Complement & increment AC	1
STL	7120	Set link to 1	1
GLK	7204	Get link (put link in AC, bit 11)	1
STA	7240	Set AC = -1	1
LAS	7604	Load AC with switch register	1

Table 5-1 (Cont)
Symbol List for TSS/8

PSEUDO-OPERATORS

DECIMAL	OCTAL
EXPUNGE	PAGE
FIELD	PAUSE
FIXTAB	TEXT
I	XLIST
	Z

Mnemonic	Code	Operation	Event Time
		IOT MICROINSTRUCTIONS	
Program Interrupt			
IOT	6000	(See Time-Sharing System User's Guide DEC-T8-MRFB-D)	
Keyboard Reader			
KSF	6031	Skip if keyboard/reader flag = 1	1
KCC	6032	Clear AC & keyboard/reader flag	2
KRS	6034	Read keyboard/reader buffer	3
KRB	6036	Clear AC & read keyboard buffer, & clear keyboard flag	2,3
KSB	6400	Set keyboard break	
SBC	6401	Set buffer control flags	
KSR	6030	Read keyboard string	
Teleprinter/Punch			
TSF	6041	Skip if teleprinter/punch flag = 1	1
TCF	6042	Clear teleprinter/punch flag	2
TPC	6044	Load teleprinter/punch buffer, Select & print	3
TLS	6046	Load teleprinter/punch buffer, Select & print, and clear Teleprinter/punch flag	2,3
SAS	6040	Send a string	
High-Speed Reader (Type PC02)			
RSF	6011	Skip if reader flag = 1	1
RRB	6012	Read reader buffer & clear flag	2
RFC	6014	Clear flag & buffer & fetch character	3
RRS	6010	Read reader string	
High-Speed Punch (Type PC03)			
PSF	6021	Skip if punch flag = 1	1
PCF	6022	Clear flag & buffer	2
PPC	6024	Load buffer & punch character	3
PLS	6026	Clear flag & buffer, load & punch	2,3
PST	6020	Punch string	

Table 5-1 (Cont)
Symbol List for TSS/8

Mnemonic	Code	Operation	Event Time
DECtape Transport (Type TU55) and Control (Type TC01)			
DTXA	6764	Load status register A	3
DTSF	6771	Skip on flags	1
DTRB	6772	Read status register B	2
Program Control			
URT	6411	User run time	
TOD	6412	Time of day	
RCR	6413	Return clock rate	
DATE	6414	Date	
STM	6415	Quantum synchronization	
TSS	6420	Skip on TSS/8	
USE	6421	User	
SSW	6430	Set switch register	
CKS	6200	Check status	
ASD	6440	Assign device	
REL	6442	Release device	
DUP	6402	Duplex	
CON	6422	Console	
File Control			
WHO	6616	Who	
SIZE	6614	Segment size	
RFILE	6603	Read file	
WFILE	6605	Write file	
ACT	6617	Account number	
REN	6600	Rename file	
OPEN	6601	Open file	
CLOS	6602	Close file	
PROT	6604	Protect file	
CRF	6610	Create file	
EXT	6611	Extend file	
RED	6612	Reduce file	
FINF	6613	File information	

5.5 ERROR DIAGNOSTICS

Table 5-2
PAL-D Error Diagnostics

Error Code	Explanation
BE	Two PAL-D internal tables have overlapped - This situation can usually be corrected by decreasing the level of literal nesting or number of current page literals used prior to this point on the page.
DE	System device error - An error was detected when trying to read or write onto the system device; after three failures, control is returned to the Monitor.
DF	Systems device full - The capacity of the systems device has been exceeded; assembly is terminated and control is returned to the Monitor.
IC	Illegal character - An illegal character was encountered other than in a comment or TEXT field; the character is ignored and the assembly continued.
ID	Illegal redefinition of a symbol - An attempt was made to give a previously defined symbol a new value by means other than the equal sign; the symbol was not redefined.
IE	Illegal equals - An equal sign was used in the wrong context. Examples: TAD A +=B — the expression to the left of the equal sign is not a single symbol or, the expression to the right of A +B=C — the equal sign was not previously defined
II	Illegal indirect - An off-page reference was made; a link could not be generated because the indirect bit was already set.
ND	The program terminator, $, is missing.
PE	Current nonzero page exceeded - An attempt was made to a. override a literal with an instruction, or b. override an instruction with a literal; this can be corrected by (1) decreasing the number of literals on the page or (2) decreasing the number of instructions on the page.
PH	Phase error - PAL-D has received input files in an incorrect order; assembly is terminated and control is returned to the Monitor.
SE	Symbol table exceeded - Assembly is terminated and control is returned to the Monitor; the symbol table may be expanded to contain up to 1184 user symbols by saving a file named .SYM on the system device.
US	Undefined symbol - A symbol has been processed during pass 2 that was not defined before the end of pass 1.
ZE	Page 0 exceeded - Same as PE except with reference to page 0.

CHAPTER 6
UTILITY PROGRAMS

6.1 EDIT

TSS/8 Editor provides the user with a powerful tool for creating and modifying source files on-line. Its precise capabilities and commands are detailed in the PDP-8 Symbolic Editor Programming Manual (Order No. DEC-08-ESAB-D). EDIT allows the user to delete, insert, change, and append lines of text; and then obtain a clean listing of the updated file. EDIT also contains commands for searching the file for a given character.

EDIT considers a file to be divided into logical units, called pages. A page of text is generally 50-60 lines long, and hence corresponds to a physical page of program listing. A FORTRAN-D program is generally 1-3 pages in length; a program prepared for PAL-D may be several pages in length. EDIT operates on one page of text at a time, allowing the user to relate his editing to the physical pages of his listing. EDIT reads a page of text from the input file into its internal buffer where the page becomes available for editing. When a page has been completely updated, it is written onto the output file and the next page of the input file is made available. EDIT provides several powerful commands for "paging" through the source file quickly and conveniently.

NOTE

The end of a page of text is marked by a form feed (CTRL/L) character. Form feed is ignored by all TSS/8 language processors.

To call the Editor, type:

```
.R EDIT
```

EDIT responds by requesting INPUT: Type and enter the name of the source file to be edited. If a new file is to be created using EDIT, there is no input file. In this case, strike the RETURN key. EDIT then requests OUTPUT: Type in the name of the new, edited, file to be created. The name of the output file must be different from the name of the input file. If EDIT is being called to list the input file, there is no need to create an output file; strike the RETURN key. When EDIT sets up its internal files and is ready for a command, it rings the bell on the Teletype.

For example:

```
.R EDIT
 INPUT:WXZOLD
OUTPUT:XYZNEW
```

(Bell rings at this point.)

6.1.1 Summary of Symbolic Editor Operations

Table 6-1
Summary of Symbolic Editor Operations

Special Characters	Function
Carriage Return (RETURN Key)	Text Mode - Enter the line in the text buffer. Command Mode - Execute the command.
Back Arrow (←)	Text Mode - Cancel the entire line of text, continue typing on same line. Command Mode - Cancel command. Editor issues a ? and carriage return/line feed.
Rubout (\)	Text Mode - Delete from right to left one character for each rubout typed. Does not delete past the beginning of the line. Is not in effect during a READ command. Command Mode - Same as back arrow.
Form Feed (CTRL/FORM Combination)	Text Mode - End of inputs return to command mode.
Period (.)	Command Mode - Current line counter used as argument alone or in combination with + or - and a number (.,.+5L).
Slash (/)	Command Mode - Value equal to number of last line in buffer. Used as argument (/-5,/L).
Line Feed (↓)	Text Mode - Used in SEARCH command to insert a CR/LF combination into the line being searched.
Right Angle Bracket (>)	Command Mode - List the next line (equivalent to .+1L).
Left Angle Bracket (<)	Command Mode - List the previous line (equivalent to .-1L).
Equal Sign (+)	Command Mode - Used in conjunction with . and / to obtain their value (.=27).
Tabulation (CTRL/TAB Key Combination)	Text Mode - Produces a tabulation which, on output, is interpreted as spaces if bit 1 of the switch register is set to 0, or as a tab character/rubout combination if bit 1 is set to 1.

6.1.2 EDIT Command Summary

Table 6-2
EDIT Command Summary

Command	Format(s)	Meaning
READ	R	Read text from the input file and append to buffer until a form feed is encountered.
APPEND	A	Append incoming text from keyboard to any already in buffer until a form feed is encountered.
LIST	L	List the entire buffer.
	nL	List line n.
	m,nL	List lines m through n inclusive.
PROCEED	P	Output the contents of the buffer to the output file, followed by a form feed.
	nP	Output line n, followed by a form feed.
	m,nP	Output lines m through n inclusive followed by a form feed.
TERMINATE	T	Close out the output file and return to TSS/8 Monitor.
NEXT	N	Output the entire buffer and a form feed, kill the buffer and read the next page.
	nN	Repeat the above sequence n times.
KILL	K	Kill the buffer (i.e., delete all text lines).
DELETE	nD	Delete line n of the text.
	m,nD	Delete lines m through n inclusive.
INSERT	I	Insert before line 1 all the text from the keyboard until a form feed is entered.
	nI	Insert before line n until a form feed is entered.
CHANGE	nC	Delete line n, replace it with any number of lines from the keyboard until a form feed is entered.
	m,nC	Delete lines m through n, replace from keyboard as above until form feed is entered.
MOVE	m,n$kM	Move lines m through n inclusive to before line k.
GET	G	Get and list the next line beginning with a tag.
SEARCH	S	Search the entire buffer for the character specified (but not echoed) after the carriage return. Allow modification when found. TSS/8 Editor outputs a slash (/) before beginning a SEARCH.

Table 6-2 (Cont)
EDIT Command Summary

Command	Format(s)	Meaning
SEARCH (Cont)	nS	Search line n, as above, allow modification.
	m,nS	Search lines m through n inclusive, allow modification.
END	E	Output the contents of the buffer. Read in any pages remaining in the input file, outputting them to the output file. When everything in the input file has been moved to the output file, close it out and return to the TSS/8 Monitor. E is equivalent to a sufficient number of N's followed by a T command.
↑C	CTRL/C	Stop listing and return to Command Mode.

6.2 LOADER

TSS/8 LOADER is used to load programs in BIN format from a disk file into the user's core area for execution. These files in BIN format can be created by PAL-D in the course of an assembly or they can be loaded from paper tape using PIP (see the PIP writeup for special instructions on loading BIN format tapes).

To call LOADER, type:

```
.R LOADER
```

LOADER responds by asking for INPUT: Respond by entering the name of the file or files to be loaded. Although many System Library Programs allow multiple input files, the LOADER uses this feature to special advantage. Because it loads the files in the order they are typed, LOADER can be used to load patches and overlays. After it has requested INPUT, LOADER requests OPTION: For normal operation strike the RETURN key; LOADER is able to load into any part of core below 7750. If the program to be loaded is to be debugged, respond to OPTION: with D. This will cause ODT to be loaded along with the input files and started. ODT indicates that it is ready by printing a second line feed. ODT uses locations 7000 through 7577; and if loaded along with a program which uses any of these locations, the result of the load is unpredictable.

Example 1: Normal Operation

```
.R LOADER

INPUT: MAIN, PATCH1, PATCH2
OPTION:
↑BS
.
```

Example 2: Load ODT with Input File

```
.R LOADER

INPUT:PROG1
OPTION: D
```

As seen in the first example, LOADER returns control to Monitor when it is finished. The user can then start the program by using the Monitor command START. For example, LOADER can be used to load and run the short program given as an example in the section on PAL-D (see Chapter 5, Section 5.3).

```
.R LOADER

INPUT: BIN2
OPTION:
↑BS
.START 400


0123456789
↑BS
.
```

NOTE

The BIN format files loaded by LOADER include a checksum. If LOADER detects a checksum error while loading, it types "LOAD ERROR" and terminates the load.

6.3 ODT (Octal Debugging Technique)

ODT is a powerful octal debugging tool for testing and modifying PDP-8 programs in actual machine language. It allows the user to control the execution of his program and, where necessary, make immediate corrections to the program without the need to reassemble.

The complete command repertoire of ODT is documented in the ODT Manual (Order No. DEC-08-COCO-D). ODT (on TSS/8) is the high-core version which resides in locations 7000 through 7577. The paper-tape output commands of regular ODT are not available in TSS/8 ODT. To call ODT, type:

```
.LOAD 2 ODTHI 0 7000
.START 7000
```

If ODT is to be used to debug a program being loaded with LOADER, ODT can be loaded and started directly by specifying the Debug (D) option to LOADER.

6.3.1 Programming Notes

ODT executes an SRA (Set Restart Address) as part of its initialization process. As a result, typing CTRL/C always returns control to ODT. If the program being debugged sets up its own restart address, typing CTRL/C transfers control to the new restart address. It is necessary to type ↑BS followed by START 7000 to force control back to ODT.

Every time ODT regains control, it puts the Teletype in duplex mode. Users debugging programs which do not operate in duplex mode, should be aware of this fact.

ODT saves the state of the delimiter mask, when it regains control via a breakpoint. The state of this mask is restored on a Continue (C) command, but not on a GO (G) command.

6.3.2 ODT Command Summary

Table 6-3
ODT Command Summary

Command	Meaning
nnnn/	Open register designated by the octal number nnnn.
/	Reopen latest opened register.
RETURN	Close previously opened register.
LINE FEED (↓)	Close register and open the next sequential one for modification.
Up Arrow (↑) (SHIFT/N)	Close register, take contents of that register as a memory reference and open it.
Back Arrow (←) (SHIFT/O)	Close register open indirectly.
Illegal Character	Current line typed by user is ignored, ODT types ?CR/LF.
nnnnG	Transfer program control to location nnnn.
nnnnB	Establish a breakpoint at location nnnn.
B	Remove the breakpoint.
A	Open for modification, the register in which the contents of AC were stored when the breakpoint was encountered.
C	Proceed from a breakpoint.
nnnnC	Continue from a breakpoint and iterate past the breakpoint nnnn times before interrupting the user's program at the breakpoint location.
M	Open the search mask register, initially set to 7777. It may be changed by opening the search mask register and typing the desired value after the value typed by ODT, then closing the register.
↓ (line feed)	Close search mask register and open next register immediately following, containing the location at which the search begins. It may be changed by typing the lower limit after the one typed by ODT, then closing the register.

Table 6-3 (Cont)
ODT Command Summary

Command	Meaning
↓ (line feed)	Close lower search register, open next register containing the upper search limit initially set to 7000 or 1000 (location of ODT). It may be changed by typing the desired upper limit after the one typed by ODT and closing the register with a carriage return.
nnnnW	Search the portion of core as defined by the upper and lower limits for the octal value nnnn.

6.4 CAT

TSS/8 Monitor maintains a library of disk files for each user. The System Library Program CAT is used to obtain a catalog of the contents of this library. For each file, CAT types the size of the file in units of disk segments. The size of a disk segment may vary among installations. Generally, it is 256 (decimal) words of disk storage. The protection code for the file is also given. (See the section on the PROTECT Monitor command (Chapter 9) for a precise explanation of protection codes.) If the program was created by any of the System Library Programs, it has a protection code of 12, meaning that other users can read the file, but only the owner can change it. To call CAT, type:

```
.R CAT
```

6.4.1 Example of CAT Usage

```
.R CAT

DISC FILES FOR USER  312    14 NOV 69

 NAME    SIZE     PROT        DATE


BAS016     1       12       14 NOV 69
BAS116     1       12       14 NOV 69
PRIME      8       12       14 NOV 69
MATRIX     3       12       14 NOV 69
SOLVE      1       12       14 NOV 69
PROG      11       12       14 NOV 69
TYPE2      2       12       14 NOV 69
BIN2       1       12       14 NOV 69
TEMP27     15      12       14 NOV 69

TOTAL DISC SEGMENTS:  43

↑BS
.
```

6.5 SYSTAT (System Status)

It is frequently useful to know the status of the system as a whole; how many users are on-line, where they are, what they are doing, etc. The SYSTAT program provides this capability. To call SYSTAT, type:

```
.SYSTAT
```

or

```
.R SYSTAT
```

SYSTAT responds by printing on the first line: the version of the TSS/8 Monitor being run, the time, and the date. SYSTAT reports the uptime which is the length of time in hours, minutes, and seconds since the system was last put on-line.

SYSTAT lists all on-line users. Each user is identified by his account number. The job number assigned to him and the number of the console he is using are indicated, as is the particular System Program he is running. The exact running state of each user, whether he is actually running (RUN), typing in (KEY) or out (TTY), doing input/output on another system device (IO or FIP), or not running (↑B), is indicated. The amount of computer time used by each user since he logged in is given.

If more users are on-line than the system has core fields to hold them, the fact that the system is swapping is reported. The number of free core blocks used internally by TSS/8 Monitor for Teletype buffering and various other purposes is typed out. Then SYSTAT reports any unavailable devices, i.e., devices which are assigned to individual users. The job to which they are attached and their status (AS if they are assigned but not active, AS+INIT if they are assigned and active) is also indicated. Finally, the number of available segments of disk storage is reported.

6.5.1 Example of SYSTAT Usage

```
.R SYSTAT

STATUS OF TSS/8.19 DEC PDP-8 #1 AT 10:27:33 ON 13 OCT 69
UPTIME 84:12:33

JOB     WHO     WHERE   WHAT    STATE   RUNTIME

1       5440    K00     SYSTAT  RUN     01:00:12
2         10    K14     COPY    KEY     00:00:21
3       3141    K15     EDIT    KEY     00:00:03
4       4771    K16     FORT    RUN     00:09:42
5       4772    K17     PALD    TTY     00:00:13
7        333    K04     BASIC   IO      01:37:12
11         7    K21     FOCAL   RUN     00:00:57
12      1221    K06     BASIC   ↑B      00:03:51
14      2000    K03     PIP     RUN     00:16:23
15       222    K12     FOCAL   KEY     00:22:17
17      5440    K11     FORT    TTY     00:22:19

SWAPPING  28K  FREE CORE=238
```

```
BUSY DEVICES

DEVICE  JOB     WHY

R       1       AS+INIT
P       3       AS
DØ      3       AS
D5      2       AS+INIT

412 FREE DISC SEGMENTS


↑BS
.
```

CHAPTER 7

PROGRAMS FOR PAPER TAPE AND DECTAPE CONTROL

7.1 PIP (Peripheral Interchange Program)

All TSS/8 System Library programs discussed in previous sections operate only on files which are on the disk. Disk is a convenient storage medium for many files; however, it may be more useful to keep some programs on paper tape. PIP provides a convenient means of transferring files between disk and paper tape, for those users who wish to preserve copies of their files off-line.

7.1.1 PIP Conventions

PIP may be considered a link between disk file storage and paper-tape devices. To punch out a desired file, PIP obtains that file from the disk and punches it on paper tape. Similarly, to load a paper tape, PIP inputs the tape from the reader, then outputs it to a disk file.

The way files are named is important to PIP. Files on disk are always named. Paper tapes, on the other hand, have no names as far as the system is concerned (although the user can label the physical tape in any manner he chooses). Paper tapes never have file names; therefore, PIP uses the absence of a file name to indicate a paper tape (absence of a file name is indicated by striking the RETURN key).

The way in which INPUT: and OUTPUT: is indicated provides the means for determining the direction of file transfer. If PIP is to get its input from the disk, the input is a file name; if the input is from a paper tape no file name is given. Similarly, if PIP is to output to the disk, the file name is indicated; if output is to paper tape, no name is given. To call PIP, type:

```
.R PIP
```

7.1.2 Using PIP to Load a Paper Tape to a Disk File

To move a paper tape to disk, strike the RETURN key when PIP requests INPUT: Since PIP must output to the disk, respond to OUTPUT: by typing a file name. When PIP requests OPTION: type T to indicate that the

paper tape is being loaded from the Teletype reader. For example:

```
.R PIP

INPUT:
OUTPUT: FILE1
OPTION:T
```

The paper tape, in the low-speed reader, is read in and stored in the system as FILE1.

7.1.3 Using PIP to Punch Out a Disk File

To move a disk file onto paper tape, the use of file names is reversed since PIP must input a disk file and output it to paper tape. The option remains the same. For example:

```
.R PIP

INPUT:FILE1
OUTPUT:
OPTION:T
```

The contents of FILE1 are then punched out at the Teletype.

7.1.4 Using PIP with the High-Speed Reader and Punch

PIP can also be used with high-speed paper-tape devices. The format of the INPUT: and OUTPUT: responses is the same. However, for the high-speed reader, the option is R and for the punch it is P.

Since the reader and punch are assignable devices, they are not always available (other users may have one or both assigned). Therefore, whenever PIP is given a command which utilizes one of these devices, it checks to make sure that the device is available. If it is, PIP automatically assigns it (thus, it is not necessary to assign the device before running PIP). If the device is unavailable, PIP so informs the user. For example:

```
INPUT:
OUTPUT: ABCD
OPTION:R
```

PIP reads the paper tape in the high-speed reader and stores it in the system as ABCD.

```
INPUT:ABCD
OUTPUT:
OPTION:P
```

PIP punches out file ABCD on the high-speed punch.

```
INPUT:ABCD
OUTPUT:
OPTION:P
DEVICE NOT AVAILABLE
```

The punch is assigned to another user, or there is no punch on the TSS/8 system, or there is one but it is turned off.

7.1.5 Using PIP to Transfer BIN Format Files

The examples above work for all ASCII file transfers (except BASIC programs, explained below.) They are also valid for punching out BIN format files with either high- or low-speed devices. Loading BIN format tapes, however, is a special case.

BIN format tapes must end with trailer codes. The easiest way to ensure that they do is to cut off the tape near the end of the trailer code. Failure to do this (or cutting it off very unevenly) does not prevent PIP from loading tape into the disk file. However, later attempts to load the file with LOADER will result in load errors.

NOTE

Some TSS/8 installations will not allow any BIN format tapes to be loaded from the low-speed reader.

7.1.6 Moving Disk Files

PIP can be used to move the contents of one file into another. This is often useful in copying a file from another user's library (providing the file is not protected) into your own library. To copy from disk file to disk file, specify a file name for both input and output. Reply to OPTION: by striking the RETURN key. For example:

```
INPUT:FOCAL 2
OUTPUT:FOCALX
OPTION:
```

PIP gets FOCAL from account number 2's library and moves it into the file FOCALX.

7.1.7 Deleting Disk Files

One of the principal reasons for punching out files on paper tape is to free disk space. Once punched out, the disk file is no longer needed. PIP offers a convenient means of deleting files, the Delete option:

```
INPUT:ABCD
OUTPUT:
OPTION:D
```

PIP deletes file ABCD, provided that the file is not protected against being changed.

7.1.8 Transferring BASIC-8 Files

BASIC-8 stores its programs in a unique file format. Therefore, it is not possible to load or punch BASIC-8 files in the usual way. To provide a convenient means of handling BASIC-8 programs, the B option is available in PIP.

The B option is used for both reading and punching BASIC-8 programs. The responses to INPUT: and OUTPUT: indicate the direction of the transfer; the high-speed reader or punch is always assumed for the B option. (To read or punch tapes at low-speed, use BASIC-8 itself.)

PIP assumes that any BASIC-8 tapes it loads are clean and error-free. Only tapes actually created by BASIC should be loaded with PIP. Tapes created off-line, and thus liable to contain errors, should be loaded low-speed by BASIC-8 itself with the TAPE command.

7.1.9 Transferring SAVE Format Files

Another special TSS/8 file format is that of the SAVE files, those programs directly executed by TSS/8. (The System Library Programs are examples of SAVE format files.) PIP provides the S option, to allow these files to be punched on paper tape. SAVE format tapes make sense only to TSS/8 PIP. They cannot be input to any other System Program.

The responses to INPUT: and OUTPUT: indicate the direction of the transfer; the high-speed reader or punch is always assumed for the S option.

NOTE

SAVE format tapes include a checksum. If PIP detects an incorrect read, it prints LOAD ERROR, and terminates the load, repeating the request for input.

7.1.10 Summary of PIP Options

Option	Explanation
T	Transfer a file between the disk and the Teletype reader or punch. The response to INPUT: and OUTPUT: indicates the direction of the transfer.
R	Read a tape from the high-speed reader and store it as a disk file.
P	Punch out the contents of a disk file on the high-speed punch.
D	Delete the file specified for input.
B	Transfer a BASIC-8 program file between the disk and the high-speed reader or punch. The response to input and output indicates the direction of the transfer.
S	Transfer a SAVE format file between the disk and the high-speed reader or punch. The response to INPUT: and OUTPUT: indicates the direction of the transfer.

7.2 COPY

Many TSS/8 installations include one or more DECtapes. For these installations, DECtape provides a convenient and inexpensive means of file storage. The COPY program is used to transfer files between disk and DECtape.

7.2.1 Using and Calling COPY

COPY is the intermediary between disk and DECtape. To write a disk file out to DECtape, COPY inputs the file from the disk, then outputs it to the DECtape. To bring a DECtape file onto the disk, COPY inputs from the DECtape, then outputs to the disk.

Files kept on DECtape have file names just as they do on the disk. To avoid confusion, the user must tell COPY where the file is to be found. If it is on DECtape, the DECtape designation and the number of the DECtape unit must preface the file name. The DECtape number is always separated from the file name by a colon. Thus D1:FILE1 means the file name FILE1 on the DECtape which is currently mounted on DECtape unit number one. The number of available tape units varies among installations. The maximum is eight, (numbered 0 - 7). If a file name is not prefaced by a DECtape number, then this file is assumed to be on the system disk.

Files stored on DECtape do not have protection codes in the sense that disk files do. They are, however, protected against unauthorized access. When a DECtape is not mounted, it is not available to any user. When it is mounted, it is available only to the user who has assigned the DECtape unit on which it is mounted. Even then it can not be altered unless the DECtape unit is set to WRITE ENABLE.

Users should be sure to assign a DECtape unit before mounting their tape, and dismount the tape before releasing the device. Normally, the DECtape unit to be used should be assigned before calling COPY.

```
.ASSIGN D 4
D 4 ASSIGNED
.R COPY
```

To call COPY, type:

```
.R COPY
```

COPY responds by asking which option the user wishes to employ. The COPY options are discussed below.

7.2.2 Loading Files from DECtape

To load a file onto the disk from DECtape, use the COPY option. When COPY requests OPTION- respond with COPY, or C, or strike the RETURN key (the COPY option is assumed). When COPY requests INPUT- type the number of the DECtape unit on which the file can be found (D0, D1, D2, D3, D4, D5, D6, or D7) followed by a colon and the name of the file on the DECtape. When COPY requests OUTPUT- type and enter the name to be given to the output file on the disk. COPY then moves the DECtape file onto the disk. (When using COPY, it is not mandatory to insert a space between the device designator and the device number.) For example:

```
OPTION- COPY
INPUT - D4:PQR
OUTPUT - PQR
```

If for any reason, COPY cannot find the DECtape file specified for input (the specified DECtape is unavailable or nonexistent, or the file name does not exist on that DECtape), COPY prints a ? and repeats the request for input. If the disk file specified for output already exists, COPY prints a ? and repeats the request for output. COPY does not overwrite an existing file. For example:

```
OPTION - C
INPUT - D9:PQR
?INPUT - D4:PQRS
?INPUT - D4:PQR
OUTPUT - FILE1
?OUTPUT- PQR
```

7.2.3 Saving Disk Files on DECtape

Saving a disk file on DECtape is very similar to loading one. The option is still COPY. For input, respond with the name of the file on the disk. For output, type the DECtape unit number, colon, and the name to be given to this file. For example:

```
OPTION - C
INPUT - ABCD
OUTPUT - D4:ABCD
```

If COPY cannot find the file on the disk, or if it is protected, COPY prints a ? and repeats the request for input. If COPY cannot set up the desired DECtape file (the specified DECtape does not exist or is unavailable, or it is not WRITE ENABLED, or a file by that name already exists on the tape) COPY types a ? and repeats the request for output.

7.2.4 Listing Directories

COPY can be used to list the directory of a device. To list a directory, respond to OPTION-by typing LIST, or just L. COPY then asks which device it is to list. To list a DECtape's directory, respond with the device name (D0,...,D7). Do not follow it by a colon. For example:

```
OPTION - LIST
DEVICE - D0

1298. FREE BLOCKS

NAME     SIZE     DATE
XCOPY      32    6-NOV-69
XBASIC     66    6-NOV-69
FOO        66    6-NOV-69
```

The unit of DECtape storage is the block, which is 128 (decimal) words. Because the unit of disk storage, the segment, is generally 256 words, a file occupies twice as many blocks of DECtape storage as it did segments on the disk.

COPY can also be used to list the user's disk directory. Use the LIST option, but respond to DEVICE- by simply striking the RETURN key. The directory listing is similar to the listing obtained by running the CAT program.

7.2.5 Deleting Files

COPY can be used to delete files, either on the disk or on a selected DECtape. To delete a file, respond to OPTION- by typing DELETE, or just D. Respond to INPUT- by typing the name of the file to be deleted.

If the file is on a DECtape, preface the file name with the DECtape unit number and a colon. For example:

```
OPTION - DELETE
INPUT - D4:ABCD
```

If COPY cannot find the file to be deleted, or having found it, cannot delete it (it is a protected disk file or a DECtape file on a unit which is not WRITE ENABLED), COPY prints a ? and repeats the request for INPUT-.

7.2.6 Deleting All Files on a Device

COPY can be used to delete all existing files on a device. To do so, respond to OPTION- by typing ZERO, or just Z. When COPY requests INPUT- respond with the name of the device. To delete all files on the disk, strike the RETURN key. The ZERO option should also be used to format a blank DECtape before attempting to copy any files onto it. For example:

```
OPTION - ZERO
INPUT - D4
```

or

```
OPTION - Z
INPUT -
```

COPY cannot delete files from a DECtape unless it is WRITE ENABLED. It cannot delete disk files which are write protected.

7.2.7 Summary of COPY Options

Option	Explanation
COPY	Transfer a file between disk and DECtape.
LIST	List a directory.
DELETE	Delete a file.
ZERO	Delete all files.

7.2.8 Example of COPY Usage

```
.ASSIGN D 5
D 5 ASSIGNED
.R COPY


OPTION - ZERO
DEVICE - D5


OPTION - LIST
DEVICE - D5



1462. FREE BLOCKS

NAME    SIZE    DATE


OPTION - LIST
DEVICE -


DISK FILES FOR USER 5440 ON 13-OCT-69.

NAME    SIZE    PROT       DATE
SOLVE      1      12     13-OCT-69

TOTAL DISK SEGMENTS: 1

OPTION - COPY
INPUT - SOLVE
OUTPUT - D5: SOLVE


OPTION - DELETE
INPUT - SOLVE


OPTION - LIST
DEVICE - D5



1460. FREE BLOCKS

NAME    SIZE      DATE
SOLVE      2    13-OCT-69
```

```
OPTION - LIST
DEVICE -

DISK FILES FOR USER 5440 ON 13-OCT-69.

NAME    SIZE    PROT    DATE

TOTAL DISK SEGMENTS: Ø

OPTION- COPY
INPUT- D5:SOLVE
OUTPUT- ABCD


OPTION- LIST
DEVICE-


DISK FILES FOR USER 5440 ON 13-OCT-69.

NAME    SIZE    PROT       DATE
ABCD       1      12     13-OCT-69

TOTAL DISK SEGMENTS: 1

OPTION- ↑BS
.RELEASE D 5
.
```

CHAPTER 8

ADVANCED MONITOR COMMANDS

8.1 INTRODUCTION

Chapter 2 described the fundamental Monitor commands, i.e., those needed to utilize existing TSS/8 System Library Programs. The TSS/8 Monitor also provides powerful commands for users who wish to create their own Library Programs.

To use the System Library Programs described in Chapters 3 through 7, it was not necessary to be familiar with the actual machine that runs them, the PDP-8/I. To create new Library Programs for TSS/8, this is necessary because they are written in the PDP-8 assembly language. The user codes his programs for a 4K PDP-8, subject to the time-sharing conventions discussed in the following chapter. The programs are created with TSS/8 EDIT, then assembled by PAL-D and loaded by LOADER. Only at this point are they able to be run by TSS/8. In the course of this program development, the same program exists in many formats. The source program is a disk file containing ASCII characters in an Editor format. PAL-D reads the file and translates it into a second file, the assembled program in BIN format. Neither of these files is capable of being executed directly by TSS/8. The BIN format tape must be loaded into core by LOADER before it can actually be executed.

At this point it is possible to save the program in a file format that is directly executable by TSS/8. Such a file, referred to as a SAVE format file, contains an image of the user's core area after the program has been loaded by LOADER. These SAVE format files differ from all the files which are created by System Library Programs and cannot be executed directly by TSS/8. Thus, it is not possible to save a BASIC program, (for example FILE1, while running BASIC) then return to Monitor, type R FILE1, and get meaningful results. The program in FILE1 must be executed under control of the BASIC language processor. Only SAVE format files can be called into execution directly by the R command. All TSS/8 System Library Programs are stored in SAVE format.

NOTE

In the following examples, Sn, Cn, and Dn are used to stand for alphanumeric strings (such as file names), octal numbers, and decimal numbers, respectively.

A number of Monitor command conventions are available to make the commands easier to use. First, more than one command may be typed on a line. Individual commands are separated by a semi-colon (;). Second, only enough characters of a command to uniquely specify it need be typed. Thus, DEPOSIT can be abbreviated DE or DEP.

```
.LOAD FILE1; DEP  20  7000; ST 200
```

is exactly equivalent to:

```
.LOAD FILE1
.DEPOSIT 20 7000
.START 200
```

These conventions are available for the elementary Monitor commands as well. They are, however, especially convenient for the advanced commands.

8.2 CONTROL OF USER PROGRAMS

Once a PAL-D program has been loaded by LOADER, several Monitor commands are available for controlling its execution.

Command	Explanation
START C1	Start execution of a user program at location C1. When a program is started, keyboard input is no longer interpreted as commands to Monitor. Input characters are passed to the running program. START C1 clears the user's AC and link.
START	Restart execution of a user program where it was interrupted (either by execution of an HLT or by ↑BS typed at the keyboard). When the START command is given, the program's state is restored.
DEPOSIT C1 C2...Cn	Deposit the octal values C2 to Cn in the locations starting at C1. DEPOSIT is used to make small octal modifications to a user program. No more than 10 decimal locations can be modified by a single DEPOSIT instruction.
EXAMINE C1	Print the octal contents of location C1.
EXAMINE C1 D1	Print the contents of D1 locations starting at C1.
WHERE	Print the present status of the user program. The user's AC, PC, and LINK are printed. If the TSS/8 processor includes the extended arithmetic element, two additional registers, the SC and MQ are printed.

It is possible to give these utility commands while a user program is running. The CTRL/B character (↑B) gets the attention of the Monitor without stopping program execution. (↑B followed by the S command stops the program.) ↑B can be used together with the WHERE command to follow program execution. After executing these commands, Monitor does not put the Teletype back into Monitor mode.

8.3 DEFINING DISK FILES

TSS/8 Monitor allows the user to save core images of his program on the disk for future use. However, before saving such a core image, the user must define a disk file in which to save it.

Disk files, like the user's core, are made up of 12-bit words. Unlike the user's core, which is always 4K in size, a file can be any size. The unit of disk file storage is the segment; in most installations a segment is 256 (decimal) words but can be from 128 to 1024 words long. Files are at least one segment long when created and grow by appending additional segments to the end of the file. In defining a file, the user first creates it, then extends it to whatever length he needs. To save a whole 4K image on a system with a segment size of 256 (decimal) words, a 16 segment file is required. If only part of the contents of the user's core is to be saved, a correspondingly smaller file can be used.

A file can be created at any time. However, to modify or redefine it in any way, the file must be open. Up to four files can be open for a user simultaneously. Opening a file connects it to an internal open file number (0, 1, 2, or 3). Once a file is open, it is referenced by this internal file number rather than by its file name.

8.3.1 Creating a Disk File

CREATE S1	Define a one segment area of disk space and associate with it the name given in the command line.

The file name can be one to six alphanumeric characters of which the first must be a letter. Creating a file deletes any existing file of the same name, unless that file is write protected. When created, files are always one segment in size. A new file is arbitrarily assigned a protection code of 12, meaning that other users may access it but only the owner may change it. Until it has been written in, the contents of a newly defined file are undefined.

8.3.2 Opening and Closing a File

To use a file, it must first be opened. A file can be opened on any of four internal file numbers: 0, 1, 2, or 3.

OPEN C1 S1	Associate the file S1 with the internal file number C1.

A user can have up to four files open at a time. If a file is open on an internal file number for which a file is already open, that file is first closed. For example:

```
.CREATE  AB
.OPEN 1  AB
```

AB is now an open file and can be referenced as file 1.

An open file can be closed at any time by means of the CLOSE command.

CLOSE C1	Close the file presently open on internal file number C1.

Once closed, a file cannot be accessed in any way until it is reopened. It is possible to close more than one file with a single command.

```
.CLOSE Ø 1 2 3
```

8.3.3 Extending, Reducing, and Renaming a Disk File

When created, a file is one segment long. If a larger file is needed, the original file can be extended.

EXTEND C1 D1	Extend the file presently open on internal file C1 by D1 segments.

Extending a file adds one or more segments to the end of that file. The contents of the old part of the file are not changed. Until written in, the contents of the newly added segments are unspecified.

An existing file may be reduced in size by means of the REDUCE command:

REDUCE C1 D1	Reduce the file presently open on internal file C1 by D1 segments.

Reducing a file deletes the number of segments indicated from the end of the file. The contents of remaining segments of the file are unchanged. If a file is reduced to zero segments, or if D1 is greater than the number of segments in the file, it is deleted entirely. An example of the creation and deletion of a 4K file:

```
.CREATE FOURK
.OPEN 3 FOURK
.EXTEND 3 15
.REDUCE 3 16
```

Existing opened files can be renamed. Renaming a file does not change its contents in any way.

RENAME C1 S1	The file open on internal file number C1 is given the new name S1.

8.3.4 Protection Codes

The user can protect his files against unauthorized access. He can also specify the extent of access certain other users can have to his files. For example, a user's associates can be permitted to look at the data of certain files but not permitted to alter that data.

When it is created, a file is assigned a protection code of 12. This protection code is defined below and can be changed (see Appendix F), but only by the owner of that file.

PROTECT C1 C2	The file open on internal file number C1 is given the protection code C2.

The protection code is actually a 5-bit mask. Each bit specifies a unique level of protection. (See the PROTECT IOT in Chapter 9 for the meaning of each bit.)

File protection masks (C2) are assigned as follows:

1	Read protect against users whose project number differs from owner's.
2	Write protect against users whose project number differs from owner's.
4	Read protect against users whose project number is same as owner's.
10	Write protect against users whose project number is same as owner's.
20	Write protect against owner. To change the program the owner must change the protect code.

Protection codes are determined as the unique sum of any of the above codes.

Some of the more common protection codes are as follows:

.PROTECT 1 12	Allow other users to access the file but not change it.
.PROTECT 1 17	Allow only the file owner to read the file. He can also change it.
.PROTECT 1 37	Allow only the file owner to read the file. He cannot, however, change it. (To change it, he must first change the protection.)
.PROTECT 1 0	Allow other users to access the file and change it.

Finally, the user can ask what file is open on a given internal file number by means of the F (File information) command.

F C1	Type out the following information about the file presently open on internal file C1: a. Account number of file owner. b. Name of file. c. Protection code. d. Size of file in segments (decimal).

For example:

```
.F 1
0010 TYPE 0012    2
.
```

8.3.5 Error Conditions

There are a number of error conditions which prevent the execution of the file definition commands (as previously described). One of the following error messages is typed by Monitor in the event that an error condition is detected:

FILE NOT OPEN	An EXTEND, REDUCE, PROTECT, or RENAME command has been issued for an internal file number for which no file is open.
FILE IN USE	An EXTEND, REDUCE, PROTECT, or RENAME command has been issued for a file which is in use elsewhere by another user. Because changing a file which is being used (i.e., has been opened) could disrupt another user's work, under these conditions such a change is prohibited.

NOTE

Changing a file which the user himself has opened on another internal file results in this error message.

DIRECTORY FULL	A CREATE command has been issued, but the user's directory is full. He can delete any of his files to make room for the new file.
PROTECTION VIOLATION	An attempt has been made to change a file which is write protected against the user.
FILE NOT FOUND	The user has attempted to OPEN a nonexistent file.
FAILED BY n SEGMENTS	The user has attempted to extend a file, but the system has run out of disk segments. The number of segments requested, but not available, is typed out.

8.4 SAVING AND RESTORING USER PROGRAMS

Once a file has been defined, the user can save all or any part of his user core in the file. Files and user core are addressed in the same way, by 12-bit words. The user can transfer his file into any part of core.

The SAVE command requires one to five parameters. The name of the file to be written into must always be given. If the file is not in the user's own library, the appropriate account number is entered before the file name. (Writing into a file owned by another user is subject to file protection.) In either case, the parameters are separated by spaces.

SAVE	Write the indicated section of core out into the indicated file.

If no parameters follow the file name, Monitor starts at location zero of the user's core and saves it in location zero of the disk file. It continues to write core locations into the disk file until: (a) it has written out the whole 4K or (b) it has filled the file. Either condition completes the SAVE.

The user can further define his SAVE command by indicating specific parts of core to be saved in specific parts of the disk file. He does this by typing one to three parameters following the file name. The first parameter following the file name indicates a specific disk file address at which to begin writing. The second parameter following the file name indicates a specific core address at which to terminate the transfer. If only the first two parameters are typed, the transfer terminates when either the end of core or the end of file is reached.

SAVE S1 SAVE C1 S1	Assuming that a disk file S1 exists, and that it is not write protected, the contents of core is saved in S1. In the first case, S1 is assumed to be in the library of the user giving the command. In the second case, it is assumed to be in the library of the user whose account number is C1.
SAVE S1 C2 C3 C4	Locations C3 to C4 (inclusive) are saved in file S1 starting at disk file location C2. S1 is assumed to be in the user's own library. If S1 is preceded by the parameter C1, it is assumed to be in the library of the user whose account number is C1.

Once a core image has been saved in a disk file, it can be restored to core by means of the LOAD command. It should be noted that the Monitor command LOAD is very different from the System Library Program LOADER. LOADER loads a BIN format file (created by PAL-D) into the user's core. LOAD loads a SAVE format file (created by a previous SAVE command) into core.

The LOAD command requires from one to five parameters. The name of the file to be loaded must always be given. If the file is in the user's own library, this file name is typed after the SAVE command itself. If it is in another user's library, his account number is entered before the file name. (Reading another user's file is subject to file protection.) In either case, the parameters are separated by spaces.

LOAD	Read the indicated section of a disk file into the indicated section of core.
LOAD S1 LOAD C1 S1	Assuming that a disk file S1 exists, and that it is not read protected, the contents of the file S1 is loaded into core. In the first case S1 is assumed to be in the library of the user giving the command. In the second case, it is assumed to be in the library of the user whose account number is D1.

The user can further define his LOAD command by using the same optional parameters discussed in the section on the SAVE command.

LOAD S1 C2 C3 C4	Locations C3 to C4 (inclusive) are loaded from file S1 starting at file location C2.

Example:

.LOAD NEWF 5 10 17	Words 5 to 14 (inclusive) of the file named NEWF are loaded into locations 10 to 17 of the user's core.

It is not necessary to open a file before using it in a LOAD or SAVE command. Both commands automatically open the specified file on internal file number 3 before performing the transfer. After completion of the command, the file remains open on file number 3.

A special macro-command, RUN, exists to allow a program to be loaded and started all in one command.

RUN S1 RUN C1 S1	Load file S1 into core from the disk and start execution at location 0. In the first example, file S1 is assumed to be in the user's own library. In the second, it is assumed to be in the library of the user whose account number is C1. RUN S1 is exactly equivalent to LOAD S1; START 0. RUN C1 S1 is exactly equivalent to LOAD C1 S1; START 0.

The R command (see Chapter 2, Section 2.4) is a special case of the RUN command.

R S1	Load file S1 from the System Library (account number 2) and start at location 0. R S1 is exactly equivalent to RUN 2 S1.

8.5 UTILITY COMMANDS

TSS/8 Monitor provides a number of special purpose commands to aid in program development and use.

Command	Explanation
USER	Print the number of the job connected with this user and the console number of the job.
USER C1	Print the console numbers of job C1.
SWITCH C1	Set the user's switch register to C1. Monitor maintains a switch register for each user. When his program executes an OSR (OR the switch register into the AC) this value is the one which is loaded.
BREAK	Type the current value of the user's delimiter mask.
BREAK C1	Set the user's delimiter mask to C1. (The use of the delimiter mask is discussed in the chapter on assembly language programming.)
DUPLEX	Place the user's Teletype in duplex mode. All characters typed at the keyboard are automatically printed as they are entered.
UNDUPLEX	Take the user's Teletype out of duplex mode. Input characters are received by the Monitor and by the user program without their being printed at the console.
RESTART C1	Set the user program restart address to C1. If CTRL/C is typed at the keyboard, Monitor forces a jump to location C1 in the user's program.
VERSION	Type out the version of TSS/8 Monitor being used.

CHAPTER 9

WRITING ASSEMBLY LANGUAGE PROGRAMS FOR TSS/8

9.1 INTRODUCTION

In addition to the higher-level programming languages available in the TSS/8 library, the user can also code and run programs written in the PDP-8 assembly language, PAL-D (Program Assembly Language). These programs are prepared with EDIT, assembled with PAL-D, then loaded with LOADER. For those users unfamiliar with assembly language programming, the DEC Introduction to Programming is a useful guide.

A user can program TSS/8 just as he would any other 4K PDP-8. (Assembly language programs must fit in 4K of core.) All memory reference instructions (AND, TAD, ISZ, DCA, JMS, and JMP) function as on a stand-alone PDP-8. All operate instructions (instruction code 7) also function as on a regular PDP-8 (Exception: microcoding HLT or OSR with any other operate instruction but CLA gives unpredictable results).

The major difference between TSS/8 programming and regular PDP-8 programming is in the IOT (Input/Output Transfer) instructions. Some instructions which are valid on stand-alone PDP-8s, such as CDF, CIF, ION, IOF are considered illegal instructions under time-sharing. There are a great many new IOTs within TSS/8 that are not valid on a regular PDP-8. Finally, there are IOTs which operate on TSS/8 in the same manner as on stand-alone PDP-8s.

The way TSS/8 actually executes an IOT instruction is also different. Non-IOT instructions (except HLT and OSR) are executed by the TSS/8 hardware, while IOTs (and HLT and OSR) are not. As explained in Appendix C IOTs are privileged instructions, executed by the TSS/8 Monitor rather than by hardware.

In general, TSS/8 provides the programming capabilities of a 4K PDP-8, and allows programs of considerably greater complexity to be run within the constraints of each user's 4K of core. The TSS/8 Library Programs, all of which were written in assembly language and make use of the TSS/8 IOT's dealt with below, are examples of programs which can be run on TSS/8.

9.2 CONSOLE I/O

User programs handle console (Teletype) I/O in almost the same way as stand-alone PDP-8 programs. The KRB instruction is used to input a character, the TLS instruction to output a character. The KSF and TSF (followed by JMP .-1) can be used but are not needed. TSS/8 Monitor handles all timing problems whether these skip IOTs are present or not.

TSS/8 differs from the stand-alone PDP-8 in that under TSS/8 the user program interacts with multi-character input and output buffers (maintained by TSS/8 Monitor) rather than with single character registers. Depending on the state of the system, these buffers may have one, many, or no characters in them. During normal program execution, this fact is of no consequence. User programs still send and receive characters one at a time. There are times, however, when it is useful to clear out any and all characters in the buffers; a special IOT exists for this purpose (SBC).

On a stand-alone system, characters are input as soon as they are typed, whether they are of immediate interest or not. Usually, these characters are stored away by the program until a terminating (or delimiting) character is found. At this time, the whole line of characters is processed. On a swapping, time-sharing system such as TSS/8, this mode of operation is wasteful. It is far more efficient to allow input characters to accumulate in the Monitor input buffer until a delimiter is found. There is an IOT to specify which characters are to be considered delimiters (KSB).

TSS/8 also allows programs to input and output strings of characters. The read string (KSR) and send string (SAS) instructions provide a convenient and efficient means of doing lengthy transfers.

All keyboard input uses full-duplexed hardware, (there is no wired connection between the keyboard and printer, i.e., characters are not printed on the console as typed). Input characters are echoed to the console under program control rather than by hardware. Because input characters are allowed to accumulate in buffers before being passed to the user program, it is important to have Monitor perform the echoing rather than user programs. There is an IOT (DUP) to set up this automatic echoing as well as an IOT (UND) to inhibit echoing for such operations as reading tapes.

Read Keyboard Buffer (KRB) — Octal Code: 6036

Operation: Read the next input character into bits 4-11 of the AC.

Load Teleprinter Sequence (TLS) — Octal Code: 6046

Operation: The ASCII character in AC bits 4-11 is typed out on the user's console.

Skip on Keyboard Flag (KSF) — Octal Code: 6031

Operation: The next instruction is skipped if there is a delimiter character in the user's input buffer.

Read Keyboard String (KSR) Octal Code: 6040

Operation: Execution of this instruction initiates a transfer of one or more characters from the user's keyboard to a designated core area. Before executing KSR, load the AC with the address of a two-word block, where:

Word 1: negative of the number of characters to be transferred.

Word 2: address of the core area into which characters are to be placed minus one.

The transfer is terminated when either:

a. the indicated number of characters have been input or

b. a delimiter is seen. At the end of the transfer, the word count and core address are updated and the AC is cleared.

Send A String (SAS) Octal Code: 6040

Operation: Before executing an SAS, load the AC with the address of a two-word block, where

Word 1: contains the negative of the number of characters to be sent.

Word 2: contains the address -1 of the first word of the string.

The characters are stared one per word right justified starting at the address specified by word 2. Upon execution of SAS, the system takes only as many characters as will fit in the output buffer. It then makes the appropriate adjustment to word 2 to indicate a new starting address and to word 1 to indicate the reduced character count; it returns to the instruction following the SAS. If the character count is reduced to zero, the instruction following SAS is skipped. The instruction following the SAS should contain a JMP .-2 to continue the block transfer of Teletype characters. The AC is cleared by SAS.

Set Keyboard Break (KSB) Octal Code: 6400

Operation: Rather than activate a user's program to receive each character as it is typed, TSS/8 accumulates input characters until a certain character or characters is seen. To tell TSS/8 Monitor which characters to look for (these characters are referred to as delimiters,) load the AC with a 12-bit mask before executing a KSB. For each bit in the mask which is set, Monitor considers the corresponding character or characters to be delimiters.

Bit	Specifies
0	0 = check rest of mask 1 = any character is break
1	301-332 (all letters)
2	260-271 (all numbers)
3	211 (Horizontal tab)
4	212-215 (line feed, vertical tab, form feed, RETURN)
5	241-273 (! " # $ % & ' ( ) * + , - . / : ;)
6	240 (space)
7	274-300 (< = > ? @)
8	333-337 ([\ ] ↑ ←)
9	377 (rubout)
10	375 (alt mode)
11	anything not in bits 1-10

Duplex (DUP) Octal Code: 6402

Operation: DUP informs Monitor that the user wishes each character typed at the console to be echoed on that console's printer as it is received by Monitor. The DUP instruction does not affect the user's registers.

Unduplex (UND) Octal Code: 6403

Operation: UND informs Monitor that the user wishes to suppress character echoing. This can be done for reasons of privacy or because a program does its own character echoing. The user's registers are unaffected by UND.

Set Buffer Control (SBC) Octal Code: 6401

Operation: SBC permits the user program to clear its Teletype input and/or output buffer. Before executing SBC set bits 0 and 1 of the AC as indicated below:

Bit 0	Clear output buffer
Bit 1	Clear input buffer

9.3 FILES AND DISK I/O

All user programs can gain access to the TSS/8 disk storage. The time-sharing Monitor maintains a pool of available disk space which is allocated in units referred to as segments. (The size of a disk segment varies among TSS/8 installations. Segments may be 128, 256, 512, or 1024 words each). These segments are used to make up user files on the disk. Monitor also maintains, for each user, a directory of all files which he has defined.

The IOTs which allow the user to access the disk are of two types: those which define files on the disk and those which transfer data between a defined file and the user's core.

NOTE

CREATE and OPEN require that a user set up a file name in core. FINF and WHO return file names to core. Each must be specified in TSS/8 internal code (excess 40 code) as shown in Appendix A. Characters are packed two to a word.

The first step in defining a file is to create it. Creating a file reserves a single segment of disk storage and associates it with a name. This file can then be extended to any length desired. Extending a file appends more segments to it. Similarly, a file can be reduced by any number of segments. Reducing a file removes the last segment or segments from the file. Reducing a file to zero segments deletes it entirely. Once created, a file can be protected, thereby restricting access to it. When created, a file can be read by any user, but only the creator can write in it. This protection can be reset if desired. Finally, it is possible to rename an existing file.

None of these actions affect the contents of the file -- they only reserve space on the disk. Until it has been written in, the actual content of a file is unspecified. Extending a file does not alter the content of the file as it previously existed. Once defined, files can be used to read and write data. Any number of words (1 to 4096) can be moved from any part of the user's core to any part of a file (subject to file protection). The user program specifies a location in core and a word count. This indicates how many words are to be transferred and from (or to) where in core they are to be moved. Also specified is a disk file address indicating what part of the file is involved. This address is the address of a word in the file. Files are addressed in the same manner as core: in 12-bit words. Unlike core, however, files can be longer than 4K. To address these files provision is made for a 24-bit disk file address, containing the high-order and low-order file addresses.

File addresses are independent of any consideration of segments. The file address is meaningful only in defining files. Files can be read and written across segment boundaries without restriction. (The user cannot read or write beyond the last segment boundary.)

When it executes a file read or write IOT, the system updates the core address and word count and places an error code in the error word (see RFILE) if any error is detected. (The error word must be cleared before executing the IOT.) At the end of a successful transfer, the word count is set to zero and the core address set to the last word transferred. If the transfer cannot be completed for some reason, the word count and core address indicate how much of the transfer was successful; the error word indicates the cause of the failure. All file operations except CREATE (and OPEN) require that the file be open. Up to four files can be open at a time. The process of opening a file associates it with one of four internal file numbers (0, 1, 2, or 3). All file IOTs except CREAT and OPEN, are specified in terms of one of these internal file numbers, rather than a file name. IOTs operate on the file which is indicated by that internal file number at the time. It is therefore possible to write file handling programs which are independent of the actual file(s) they operate on.

File IOTs, that are successfully completed, return with the AC cleared. If an error was found which prohibited execution of the IOT, one of the following error codes is returned:

4000	There was no file opened on the specified internal file number.
4400	Attempting to redefine a file which is open to another user.
5000	Attempting to create a file for a user whose directory is full.
6000	File protection violation.
6400	Invalid file name.
7000	Attempting to open a nonexistent file.
7400	Disk is full.

Create a File (CRF) Octal Code: 6610

Operation: the user can request the system to create a new file of one segment. The user program provides the new name for the file. Load the AC with the beginning address of a 3-word block, where

Words 1 through 3: contain the 6-character name.

If there is some reason why the request cannot be granted, the system will return a non-zero error code in the AC. The protection code of a newly created file is 12.

Extend A File (EXT) Octal Code: 6611

Operation: To extend the length of an existing file, that file must be currently open. Load the AC with the beginning address of a 2-word block, where:

Word 1: contains the internal file number of the file to be extended.

Word 2: contains the number of segments the system should append to the file.

If for some reason the request to extend a file cannot be granted, the AC will contain 4000, 4400, 6000, or the number of segments it failed to append.

Reduce A File (RED) Octal Code: 6612

Operation: To reduce the length of an existing file, that file must be currently open. Load the AC with the beginning address of a 2-word block, where:

Word 1: contains the internal file number of the file to be reduced.

Word 2: contains the number of segments to be removed.

This request is granted unless the file to be reduced is currently opened to another user or if the file is write protected against the user.

Rename A File (REN) Octal Code: 6600

Operation: REN is used to change the name of a file. Load the AC with the address of a 4-word block where:

Word 1: contains the internal file number associated with the file whose name is to be changed.

Words 2-4: contains the new name. This name is in 6-bit characters packed two in a word.

Protect A File (PROT) Octal Code: 6604

Operation: The owner of a file can protect his file from unauthorized attempts to access it by using this instruction. Before executing PROT, load the AC with:

Bits 5 and 6	Internal file number of the reserved file to be protected.
Bit 7	Write protect against owner.
Bit 8	Write protect against users whose project number is same as owner's.
Bit 9	Read protect against users whose project number is same as owner's.
Bit 10	Write protect against users whose project number differs from owner's.
Bit 11	Read protect against users whose project number differs from owner's.

A file must be opened before it can be protected. PROT is legal only when performed by the file owner, i.e., the user who created the file. All attempts to access the file which violate any of the protection flags are considered illegal. (For further information on project numbers, see Appendix F.)

Open A File (OPEN) Octal Code: 6601

Operation: OPEN is used to associate a file with an internal file number, which is necessary because all file operations are in terms of the internal file numbers. Before executing the OPEN IOT, load the AC with the beginning address of a 5-word block, where:

Word 1:	contains the internal file number.
Word 2:	contains the account number of the owner of the file. If 0, the account number of the current user is specified.
Word 3-5:	contain the name of the file to be opened. This name is in 6-bit characters packed two to a word.

If there was another file associated with the internal file number before the execution of the OPEN IOT, it is closed automatically before the new file is associated with the internal file number.

Close A File (CLOS) Octal Code: 6602

Operation: CLOS terminates the association between files and their internal file numbers. Before executing CLOS, load the AC with a selection pattern for the internal file numbers whose associated files are to be closed. The file is closed if bit I is 1, where I = bit 0, 1, 2, or 3.

Read File (RFILE) and Write File (WFILE) Octal Codes: 6603 & 6605

Operation: Once the association of a file with an internal file number has been made, these IOTs allow the actual file reference to be made. They are illegal on a file that has not been opened (associated with an internal file number).

To read or write a file, load the AC with the address of a 6-word block, then execute the IOT. The format for the 6-word block is:

Word 1:	contains the high-order file word address.
Word 2:	contains the internal file number.
Word 3:	contains the negative of the number of words for the operation. This number is either the number of words to be read or the number of words to be written.
Word 4:	contains a pointer to the beginning address -1 of a buffer located in the user program. On a read operation this buffer receives the information from the file: on a write operation this buffer holds the information that is to be sent to the file.
Word 5:	contains the least significant 12 bits of the initial file word address to begin the operation.

Word 6: contains an error code:

0	if no error
1	if parity error
2	if file shorter than word count
3	if file not open
4	if protection violated

The read or write begins at the word specified by words 1 and 5. For example:

```
        TAD X
        WFILE
         .
         .
         .
X,      .+1
        Ø
        1
        -200
        6477
        200
```

means: write 200 (octal) words starting at word 200 of the file that is associated with internal file number one from a core area starting at location 6500.

After completion of the transfer, the word count (word 3) and core address (word 4) are updated. If an error was detected the appropriate error code is placed in word 6.

File Information (FINF) Octal Code: 6613

Operation: FINF enables a user program to determine what file, if any, is associated with an internal file number. Load the AC with the beginning address of a 7-word block, where:

Word 1: contains the internal file number for which the user program wishes information.

Words 2 through 7: contain the information that the system returns after executing FINF.

Word 2: contains the account number of the owner or zero if no file is associated with the internal file number, that is, the file is not open.

Words 3-5: contains the name of the file in 6-bit code.

Word 6: contains:

Bit-1	Means
7	write protected against owner
8	write protected against users whose project number is same as owner's
9	read protected against users whose project number is same as owner's
10	write protected against users whose project number differs from owner's

Word 6 (Cont)	Bit-1	Means
	11	read protected against users whose project number differs from owner's

Word 7: contains the number of segments which compose the file.

9.4 ASSIGNABLE DEVICES

Users can access both their own Teletypes and the disk; with the remaining system devices (referred to as the assignable devices) this is not true. One function of the TSS/8 Monitor is to ensure that device usage never conflicts. Only one user at a time can access the high-speed, paper-tape reader or punch, or any one of the DECtapes.

To ensure that only one user can access a device, TSS/8 requires that the device be assigned before it is used. After a device is assigned, it is not available until it is released by its owner.

Once assigned, the device is programmed exactly as on a stand-alone PDP-8. The RRB instruction is used to read a character from the high-speed reader; the PLS instruction is used to punch one on the high-speed punch. The skip IOTs (RFS and PSF) can be used (followed by JMP .-1) but are not necessary. For block transfers, there are two string transfer commands: RRS and PST.

The DECtape instructions have been simplified. A single instruction, DTXA, initiates the transfer of a block of data. The DTRB instruction is then used to determine if the transfer was successful. The skip instruction, DTSF, can be used (followed by JMP .-1) but is not necessary.

Executing any of the assignable device IOTs without first assigning the device gives the following results: (a) If the device is assigned to another user, the instruction is considered illegal; program execution is now terminated and an error message typed out; (b) If the device is available it is automatically assigned before execution of the IOT. The device then belongs to this user until he releases it.

Because these devices are shared by all users, Monitor must ensure that they are operable at all times. In particular, Monitor must ensure that a user is not waiting (futilely) for a device which is not available. This situation can arise when trying to use the punch when it is turned off, or when the reader has read off the end of a tape. All these conditions, referred to as "hung devices" are considered to be system errors. If the program doing the transfer has been enabled for system errors, (by executing an SEA) control transfers to the error routine indicated which must clear the error flag in the status word before continuing (see Section 9.4). If the user program has not been enabled for system errors, a hung device causes the program to be terminated and an error message is typed out.

Assign Device (ASD) Octal Code: 6440

Operation: If the device specified by the contents of the AC is available, it is assigned to the user program and the AC is cleared. Otherwise, the number of the job owning the device is placed in the AC. If the device does not exist, 7777 is returned in the AC.

4000	Paper-tape reader
4001	Paper-tape punch
4005 + N	DECtape unit N

The assignment is in effect until a corresponding REL instruction or LOGOUT.

Release Device (REL) Octal Codes: 6442

Operation: The device specified by the contents of the AC is released (providing it was owned by the user executing the REL). The AC is cleared. Releasing a device makes it available to other users.

Skip on Reader Flag (RSF) Octal Code: 6011

Event Time: 1

Operation: The reader flag is sensed, and if it contains a binary 1, the contents of the PC are incremented by one so that the next sequential instruction is skipped. The reader flag is bit 8 of status register 1, and has a value of 1 if the reader buffer is not empty.

Read Reader Buffer (RRB) Octal Codes: 6012 & 6016

Event Time: 2

Operation: The contents of the reader buffer are transferred into bits 4 through 11 of the AC and the reader flag is cleared if the reader buffer is empty. This instruction does not clear the AC. If the reader buffer is empty, the user program is dismissed until the reader flag is 1 or an end-of-tape condition is detected.

Reader Fetch Character (RFC) Octal Code: 6014

Event Time: 3

Operation: The reader flag and the Monitor reader buffer are both cleared, the reader is started to fill the Monitor reader buffer and the reader flag is set after the buffer is full or the end of tape is detected.

Read Reader String (RRS) Octal Code: 6010

Operation: This instruction initiates a transfer from the high-speed reader to a selected area in the user's core. Before executing RRS, load the AC with the address of a 2-word block, where:

Word 1:	minus the number of characters to be transferred.
Word 2:	the address of the user core area minus one.

The transfer is terminated by either of two conditions: (a) the word count is zero indicating that the required number of characters have been read or (b) the reader has read off the end of the tape (a system error condition).

In either case, the word count and core address are updated. RRS clears the AC.

Load Punch Buffer Sequence (PLS) Octal Code: 6026

Operation: The ASCII character is in AC bits 4 through 11 and is transmitted to the high-speed punch. PLS does not clear the accumulator.

Skip on Punch Flag (PSF) Octal Code: 6021

Event Time: 1

Operation: The punch flag is sensed, and if it contains a binary 1, the contents of the PC is incremented by one so that the next sequential instruction is skipped. The punch flag is bit 9 of status register 1, and has a value of 1 if the punch buffer is not full. If the punch flag is 0, the program is dismissed until the punch flag is 1.

Punch String (PST) Octal Code: 6020

Operation: PST allows a user program to punch a string of characters. Before executing PST, load the AC with the beginning address of a 2-word block, where:

Word 1: contains the negative of the number of characters to be punched.

Word 2: contains the beginning address -1 of the string to be punched; the characters should be right justified one per word.

After execution of PST, the system takes only as many characters as fit in the punch buffer; it then makes the appropriate adjustment to word 2 to indicate a new starting address and to word 1 to indicate the reduced character count. It returns to the instruction following the PST which should be a JMP .-2 to continue the transfer. If the character count is reduced to zero, the instruction following PST is skipped. The AC is cleared by PST.

Load Status Register A (DTXA) Octal Code: 6764

Operation: DTXA allows a user program to read and write records (129-word blocks) on DECtape. Load the AC with the beginning address of a 3-word block, where:

Word 1: contains:

Bit 1	Means
0-2	contains the transport unit select number,
3-5	0,
6-8 = 2	for read data function,
4	for write data function,
9-11	0.

Word 2: contains the DECtape block number.

Word 3: contains the beginning core address -1 of record buffer.

After DTXA is given, the DECtape request is placed in the DECtape request queue. After the completion of any DECtape request, the DECtape flag in status register 2 is turned on. DTXA does not update word 3. The AC is cleared by DTXA.

Skip on Flags (DTSF) Octal Code: 6771

Operation: The content of both the error flag and the DECtape flag is sampled, and if either flag contains a binary 1, the content of the PC is incremented by one to skip the next sequential instruction. If both flags are zero, the user program is dismissed until the skip is satisfied.

Read Status Register B (DTRB) Octal Code: 6772

Operation: The content of DECtape status register B is loaded into the AC by an OR transfer. The AC bit assignments are:

0	=	error flag
1	=	mark track error
2	=	end of tape
3	=	select error
4	=	parity error
5	=	timing error
6-10	=	unused
11	=	DECtape flag

9.5 PROGRAM CONTROL

There are a number of ways that the status of a running program can be changed. The program can be terminated in one of three ways: by execution of a HLT, by the user typing ↑BS to force a program halt, or by a program error which forces Monitor to terminate the program after typing out an error message.

It is also possible for the status of a running program to change without it being terminated. First, the user program can request that it handle its own program error conditions. In this case, Monitor does not terminate a job on an error; instead, it transfers control to a user error handler. This error handler then determines what the error was, by a CKS instruction and takes appropriate action. Monitor also provides the program with an interrupt key, ↑C.† If the user types a ↑C, Monitor unconditionally transfers control to a restart address. Thus, the user program can handle its own restarts.

Halt (HLT) Octal Code: 7402

Operation: This instruction is used to stop the user program and return control to Monitor. Executing HLT is equivalent to typing ↑BS followed by RETURN.

† ↑C is a control character obtained by depressing the CTRL key at the same time as depressing the C key. ↑BS is obtained by depressing control key and B together followed by S.

Set Restart Address (SRA) Octal Code: 6417

Operation: This instruction allows the user to specify an address to which control is transferred when an ↑C is typed on the user's console. Load the AC with the restart address and execute SRA. If ↑C is detected , the program's input and output buffers are cleared, the AC and Link are cleared and control goes to the restart address.

Set Error Address (SEA) Octal Code: 6431

Operation: This instruction allows the user to specify an address to which control is transferred in the event of a system error. Load the AC with an address before executing SEA. If a system error is detected, Monitor simulates a JMS to the error address. The program counter is stored in the error address and control transferred to the error address +1. AC, Link, and input/output buffers are not affected. The error code of the system error is in STR0 bits 9-11. Bit 10 of STR1 is set. The error routine must read these bits (by a CKS) to determine the cause of the error, then clear them by means of a CLS.

The only error code that occurs in the course of normal system usage is due to a hung device. This error occurs when the user attempts to use a punch not already turned on, access a DECtape not yet selected, or allows the paper-tape reader to run off the end of a tape. The error routine must release the device to clear the error condition. The illegal IOT error probably means that an assignable device IOT was executed without the device first being being assigned. Swap and file errors occur if a hardware error is detected while Monitor is swapping user programs or while reading or writing file directories. These are system malfunctions from which there is no recovery.

9.6 PROGRAM AND SYSTEM STATUS

Because TSS/8 programs run under control of a time-sharing Monitor, it is important for them to determine their status within the system and the status of the system as a whole. Several IOTs, listed below, have been defined for this purpose.

Check Status (CKS) Octal Code: 6200

TSS/8 Monitor maintains for each user a complete set of status information, his program's running status and the state of his input/output devices. This status information, stored in three words, can be accessed by a running program with the CKS instruction. Before executing a CKS, load the accumulator (AC) with the address of a three-word block. Executing CKS stores the three status words (STR0, STR1, and STR2) in the three-word block and clears the AC. Information about the status of individual devices can also be checked by the skip IOTs.

The formats of these registers are:

STR0 Bits

0	Run Bit	User program is in the run state
1	Error Enable	Program handles its own errors
2-4		Unused
5	JSIOT	System use only
6	JSIOTC	System use only
7	JSEXON	System use only
8		Unused
9-11	Error Code	System detected error condition 1 Illegal IOT 2 Swap read error 3 Swap write error 5 Disk file error 6 Hung device

STR1 Bits

0	Timer	Time is up
1	File 0	Internal file 0 is not busy
2	File 1	Internal file 1 is not busy
3	File 2	Internal file 2 is not busy
4	File 3	Internal file 3 is not busy
5	Delimiter	There is a delimiter in the input buffer
6		Unused
7	Teleprinter	Output buffer is not full
8	Reader	Character in reader buffer
9	Punch	Punch buffer is not full
10	Error	System error detected, code in bits 7 through 11 of STR0
11	Wait	Job is not waiting

STR2 Bits

0-8		Unused
9	DECtape	DECtape transfer requested
10	Error	DECtape error
11	DECtape	DECtape flag

Each user has available to him a 12-bit switch register just as he does on a stand-alone PDP-8. This switch register can be manipulated by means of the Monitor command SWITCH or under program control.

OR With Switch Register (OSR) Octal Code: 7404

Operation: The content of the user's switch register is inclusively ORed into the AC.

Set Switch Register (SSW) Octal Code: 6430

Operation: The content of the AC is stored in the user's switch register. The AC is cleared.

Assembly language programs run under control of TSS/8 Monitor. The following IOTs are defined to allow a program to determine the status of the system as a whole.

Segment Size (SIZE) Octal Code: 6614

Operation: The segment is the basic unit of on-line file storage. Different TSS/8 systems have differing segment sizes. SIZE allows a program to determine the segment size. The segment size is returned in the AC.

Segment Count (SEGS) Octal Code: 6406

Operation: The number of available disk segments is returned in the AC.

Account (ACT) Octal Code: 6617

Operation: The account number (of the job number given) is returned in the AC. If AC is 0, the account number for the current job is returned. If the requested job does not exist, zero is returned.

Who (WHO) Octal Code: 6616

Operation: The account number and password of the current job are returned to the 3-word block whose address is in the AC and the AC is cleared.

User (USE) Octal Code: 6421

Operation: Return in the AC the number of the current job.

Console (CON) Octal Code: 6422

Operation: Return in the AC the console unit number assigned to the job whose number is in the AC, if that console number does not exist, -1 is returned.

User Run Time (URT) Octal Code: 6411

Operation: Load the AC with the address of a 3-word block, where word 1 contains the number of the job for which the run time is sought. The run time is returned in the last two locations of the block. If job 0 is specified, the run time of the current job is returned. The AC is cleared.

Time-of-Day (TOD) Octal Code: 6412

Operation: Returns the value of the System Clock in military time (using a 24-hour cycle) in the two locations starting at the location of the address in the AC. The AC is cleared.

Return Clock Rate (RCR) Octal Code: 6413

Operation: The number of clock ticks per second is returned in the AC.

Date (DATE) Octal Code: 6414

Operation: Returns the date in the AC. The format of this 12-bit number is:

```
DATE=((YEAR-1964)*12+(MONTH-1))*31+DAY-1
```

Skip on TSS/8 (TSS) Octal Code: 6420

Operation: This instruction is used by programs which run under both TSS/8 and a standard PDP-8/I. Under TSS/8, the instruction following TSS is skipped and the Monitor version number is returned in the AC. On a standard PDP-8/I, the IOT has the effect of a NOP instruction.

Quantum Synchronization (SYN) Octal Code: 6415

Operation: Upon execution of this instruction, the system dismisses the user program and sets it in the run state so that it will be run again in turn. Ordinarily, this instruction is used to ensure a full time quantum to perform some critical operation.

Set Time (STM) Octal Code: 6416

Operation: The system provides a clock time for each user program. By means of this IOT, the time can be set to "fire" after a specified number of clock ticks have elapsed.

Load the AC with the time (in seconds) to prime the timer. Upon execution of the STM instruction, the system sets the time to "fire" in the specified number of seconds and turns the time bit (bit 0) in status register 1 to 0, clears the AC, and dismisses the job. After the specified time has elapsed, the system turns bit 0 back to 1, and the job is restarted.

9.7 PDP-8 COMPATIBILITY

Programming TSS/8 in assembly language is very similar to programming a stand-alone PDP-8. All instructions except the IOTs operate identically in either case. As discussed in the earlier sections of this chapter, programming such devices as the Teletype and high-speed reader/punch for TSS/8 is somewhat simpler. TSS/8 runs programs which include timing loops. It also properly executes the IOTs not mentioned in this manual, e.g., TCF, PCF, RRB, RFC, etc. Thus, programs written for stand-alone PDP-8s with Teletype and high-speed reader or punch will run on TSS/8, although generally not as efficiently as programs which are written specifically for TSS/8.

The same is not true for disk and DECtape operations because TSS/8 uses a simplified programming structure for these devices. The actual differences in coding are very small. It is a simple task to adapt previously written code for TSS/8 disk and DECtape.

There are a few standard changes which users generally make in adapting PDP-8 code to TSS/8. The TSS/8 Monitor does the echoing rather than the user program. The TLS which does the echo can be deleted and a DUP instruction added somewhere near the start of the program. Also, for efficiency, the TSS/8 delimiter capability can be used. A KSB in the program determines what the delimiters are.

Many PDP-8 programs execute a reader and punch IOT early in the program, to initialize the device, whether they are actually to be used or not. If the devices are free, they can be assigned and thus made unavailable to other users. If they are unavailable, the program terminates on an illegal IOT. Thus, it is important not to execute these IOTs randomly. If disk or DECtape is involved the actual transfer code must be altered to conform to TSS/8. (The fact that core page 37, locations 7600 through 7777, is available to TSS/8 programs is useful in making these changes. New code can be placed in the area normally reserved for the Binary Loader.)

The most difficult code to convert is that code which operates under interrupt. To be run under TSS/8, these programs must be recoded so as not to use the interrupt.

IOTs for nonexistent devices are ignored as are CDFs and CIFs to field zero. (Other CDFs and CIFs are illegal.) It must be remembered, that many IOTs have been redefined for use as special TSS/8 instructions. In all other situations, TSS/8 remains compatible with stand-alone systems whenever possible.

APPENDIX A

TSS/8 CHARACTER SET

TSS/8 accepts 8-bit ASCII characters only. ASCII is an abbreviation for USA Standard Code for Information Interchange. The acceptable characters and their 6- and 8-bit octal equivalents are listed below.

Character	6-Bit† Octal	8-Bit Octal	Character	6-Bit† Octal	8-Bit Octal
Space	00	240	@	40	300
!	01	241	A	41	301
"	02	242	B	42	302
#	03	243	C	43	303
$	04	244	D	44	304
%	05	245	E	45	305
&	06	246	F	46	306
'	07	247	G	47	307
(	10	250	H	50	310
)	11	251	I	51	311
*	12	252	J	52	312
+	13	253	K	53	313
,	14	254	L	54	314
-	15	255	M	55	315
.	16	256	N	56	316
/	17	257	O	57	317
0	20	260	P	60	320
1	21	261	Q	61	321
2	22	262	R	62	322
3	23	263	S	63	323
4	24	264	T	64	324
5	25	265	U	65	325
6	26	266	V	66	326
7	27	267	W	67	327
8	30	270	X	70	330
9	31	271	Y	71	331
:	32	272	Z	72	332
;	33	273	[	73	333
<	34	274	\	74	334
=	35	275	]	75	335
>	36	276	↑	76	336
?	37	277	←	77	337

†Used to store passwords and filenames only.

APPENDIX B

SUMMARY OF MONITOR COMMANDS

B.1 MONITOR COMMANDS

A Monitor command is a string of characters terminated by a semicolon (;), a colon (:), or a carriage return (RETURN key). Parameters of commands can be octal numbers, decimal numbers, character strings, or single letters. In the following summary, parameters are coded as follows:

C1, C2, . . .	represent octal numbers
D1, D2, . . .	represent decimal numbers
S1, S2, . . .	represent character strings
L1, L2, . . .	represent single letters

B.1.1 Logging In and Out

LOGIN C1 S1;	Request to login; C1 = user's account number S1 = user's password
LOGOUT;	Request to logout; processing and console time is typed out
TIME C1;	Request typeout of processing time; C1 = job number Without C1, current job is assumed; before logging in and without C1, time-of-day is typed out; If C1 = job 0, time-of-day is typed out.

B.1.2 Device Allocation

ASSIGN L1;	Access device; L1 = R for paper tape reader P for paper tape punch D for any DECtape unit
ASSIGN D C1;	Access DECtape unit; D = DECtape C1 = device number

RELEASE L1;	Release device; L1 = R, P (see ASSIGN L1;)
RELEASE D C1;	D = DECtape unit C1 = console or DECtape number

B.1.3 File Handling

OPEN C1 S1 C2;	Establish association between internal file number and file; C1 = internal file number S1 = file name C2 = account number
CLOSE S1;	Close files; S1 = list of internal file numbers
CREATE S1;	Create new file; S1 = name of new file
RENAME C1 S1;	Rename a file; C1 = internal file number S1 = new name of file
REDUCE C1 D1;	Reduce length of file; C1 = internal file number D1 = number of segments to be removed from end of file
EXTEND C1 D1;	Extend length of file; C1 = internal file number D1 = number of segments to be added to end of file
PROTECT C1 C2;	Protect a file; C1 = internal file number C2 = new file protection mask 1 read protect against users with different project number 2 write protect against users with different project number 4 read protect against users with same project number 10 write protect against users with same project number 20 write protect against owner or the sum of any combination
F C1;	Print out association between internal file numbers and files C1 = internal file number

B.1.4 Control of User Programs

START C1;	Execute user program; C1 = starting location
START;	Restart user program;
RESTART C1;	Set program restart address
DEPOSIT C1 C2 . . .Cn;	Store in core memory; C1 = location C2 = contents to be stored . . . Cn = location C1+n-1 n $\leq$ 10 decimal
EXAMINE C1 D1;	List specified contents; C1 = first location D1 = number of location to be listed D1 $\leq$ 10 decimal

B.1.5 Utility Commands

SAVE S1; SAVE C1 S1 C2; SAVE C1 S1 C2 C3; SAVE C1 S1 C2 C3 C4;	Save core image; C1 = owner's account number S1 = name of file C2 = file address of first word to be saved; if not specified, entire 4K is saved C3 = core address of first word to be saved; if not specified, entire core is saved C4 = core address of last word to be saved; if not specified, entire core is saved
LOAD C1 S1; LOAD C1 S1 C2; LOAD C1 S1 C2 C3; LOAD C1 S1 C2 C3 C4;	Load core image; C1 = owner's account number S1 = name of file C2 = file address of first word to be loaded; if not specified, entire 4K is loaded C3 = core address of first word to be loaded, if not specified, entire core is loaded C4 = core address of last word to be loaded; if not specified, entire core is loaded
R S1;	Run System file; S1 = name of file

B.1.6 Utility Commands (Cont)

RUN S1;	Run user file; S1 = name of file
RUN C1 S1;	Run user file; C1 = owner's account number S1 = name of file
S;	Stop execution
WHERE;	Type out contents of location counter, accumulator, link, and switch register
USER;	Type out number of the job and devices owned
USER C1;	Type out device numbers; C1 = user's account number
SWITCH C1;	Set switch register; C1 = word to be set
BREAK C1;	Set keyboard break mask; C1 = new mask
DUPLEX;	Echo typed characters on printer
UNDUPLEX;	Ignore previous DUPLEX command
TALK C1 S1;	Send a message to console C1; C1 = destination console S1 = message

APPENDIX C
SUMMARY OF IOT INSTRUCTIONS

C.1 PROGRAM CONTROL

Number	Instruction	Function
6200	CKS	Check Status
6402	DUP	Duplex Console
6403	UND	Unduplex Console
6405	CLS	Clear Status
6411	URT	User Run Time
6412	TOD	Time of Day
6413	RCR	Return Clock Rate
6414	DATE	Date
6415	SYN	Quartum Synchronization
6416	STM	Set Timer
6417	SRA	Set Restart Address
6420	TSS	Skip on TSS/8
6421	USE	User
6422	CON	Console
6430	SSW	Set Switch Register
6431	SEA	Set Error Address
6440	ASD	Assign Device
6442	REL	Release Device
7402	HLT	Halt
7404	OSR	OR With Switch Register

C.2 FILE CONTROL

Number	Instruction	Function
6600	REN	Rename File
6601	OPEN	Open File
6602	CLOS	Close File
6603	RFILE	Read File
6604	PROT	Protect File
6605	WFILE	Write File
6610	CRF	Create File
6611	EXT	Extend File
6612	RED	Reduce File
6613	FINF	File Information
6614	SIZE	Segment Size
6406	SEGS	Segment Count
6617	ACT	Account Number
6616	WHO	Who

C.3 INPUT BUFFER CONTROL

Number	Instruction	Function
6030	KSR	Read Keyboard String
6031	KSF	Skip On Keyboard Flag
6032	KCC	Clear Keyboard Flag
6034	KRS	Read Keyboard Buffer Static
6036	KRB	Read Keyboard Buffer Dynamic
6400	KSB	Set Keyboard Break
6401	SBC	Set Buffer Control Flags

C.4 OUTPUT BUFFER CONTROL

Number	Instruction	Function
6040	SAS	Send A String
6041	TSF	Skip On Teleprinter Flag
6042	TCF	Clear Teleprinter Flag
6044	TPC	Load Teleprinter and Print
6046	TLS	Load Teleprinter Sequence

C.5 HIGH-SPEED PAPER-TAPE READER AND CONTROL (TYPE PC02)

Number	Instruction	Function
6010	RRS	Read Reader String
6011	RSF	Skip On Reader Flag
6012	RRB	Read Reader Buffer
6014	RFC	Reader Fetch Character

C.6 HIGH-SPEED PAPER-TAPE PUNCH AND CONTROL (TYPE PC03)

Number	Instruction	Function
6020	PST	Punch String
6021	PSF	Skip On Punch Flag
6022	PCF	Clear Punch Flag
6024	PPC	Load Punch Buffer and Punch Character
6026	PLS	Load Punch Buffer Sequence

C.7 DECTAPE CONTROL (TYPE TC01)

Number	Instruction	Function
6764	DTXA	Load Status Register A
6771	DTSF	Skip On Flags
6772	DTRB	Read Status Register B

APPENDIX D

OFF-LINE TAPE PREPARATION AND EDITING

D.1 INTRODUCTION

To run a program on the computer, instructions and data must be fed into the computer from the input device.

The program and data are usually typed into the computer on-line or prepared with the aid of the Symbolic Editor. It is sometimes convenient to prepare the program and data off-line, that is, to punch the program and data onto paper tape using a separate machine, one not actually connected to the computer.

The Model 33 ASR Teletype can be used off-line to prepare source program tapes, to duplicate tapes, and to edit tapes previously punched in the ASCII format. (Tapes punched from the Teletype keyboard are in ASCII format.)

When the Teletype power control switch is turned to LOCAL, the unit becomes an off-line tape preparation facility. Procedures for using the Teletype off-line are listed below. The Teletype controls are described in Section 1.4.

D.2 DUPLICATING TAPES

The following is a description of the procedure used in duplicating tapes.

Step	Procedure
1	Turn TTY to LOCAL.
2	Set LSR to FREE.
3	Put original tape into LSR.
4	Depress LSP ON.
5	Depress HERE IS key to generate leader tape.
6	Set LSR to START. (New tape is punched and data is typed on printer.)
7	After the original tape is read in, depress HERE IS key to generate trailer tape.
8	Remove tapes from LSR and LSP.

D.2.1 Preparing New Program Tapes

When preparing a program tape off-line, the user should observe the same conventions of his programming language as when preparing a program on-line using Editor. The following are the manual operating procedures for off-line tape preparation.

Step	Procedure
1	Turn TTY to LOCAL.
2	Depress LSP ON.
3	Depress HERE IS key to generate leader tape.
4	Type the source program, observing the conventions of the programming language being used.

NOTE

The RETURN and LINE FEED keys must be depressed at the end of each line.

Depressing the CRTL/TAB keys perforates the tab character onto the tape, and the typewheel moves only one position to the right. When the computer reads the punched tab character on output, the typewheel tabs (a tab is usually equal to 10 spaces) to the next tabstop.

5	After the source program is punched, depress HERE IS to generate trailer tape.
6	Remove the source program tape from LSP.

D.2.2 Correcting Typing Errors

Typing errors can be corrected using the B. SP. button on the tape control and the RUBOUT key on the console keyboard. The B. SP. button backspaces the tape one column for each depression of the button, and the RUBOUT key perforates all eight channels of the tape (this perforation is ignored by the computer).

D.2.3 Editing

Punched tapes can be edited off-line. However, the user must be able to read the perforations on the tape, otherwise, off-line editing is virtually impossible. The following is a description of the process used to edit a paper tape off-line.

Step	Procedure
1	Turn TTY to LOCAL.
2	SET LSR to FREE.
3	Put tape to be edited into LSR.
4	Depress LSP ON.
5	Depress HERE IS to generate leader tape.
6	Set LSR to START.
7	Observe the printer as the program is being typed, and
8	Set LSR to STOP a few characters ahead of the text to be edited.
9	Advance the tape one character at a time by toggling the LSR control from START to STOP.

Minor Edit: Advance tape past the character(s) to be edited and use the B. SP. and RUBOUT keys to erase any incorrect characters, then type and punch corrected text.

Major Edit: Following Step 9:

a. Set LSR to STOP one character ahead of the text to be edited;
b. Type new text;
c. Set LSR to FREE;
d. Advance tape past edited area (reading the perforated tape);
e. Set LSR to START.

10	Repeat from step 6 until editing is completed.
11	Set LSR to START.
12	After new source program tape is punched, depress HERE IS to generate trailer tape.
13	Remove old tape from LSR and discard.†
14	Remove new tape from LSP and save.

D.2.4 Listing of a Punched Paper Tape

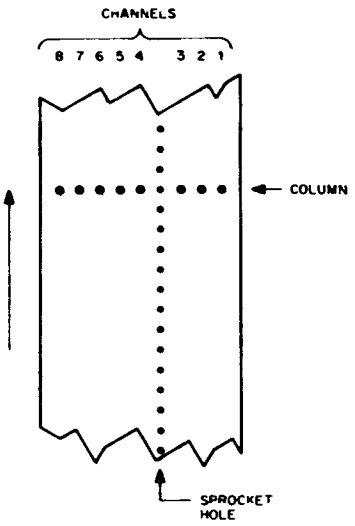
The printer can also be used to obtain a listing of a punched paper tape; this can be done with a minimum of

†It is good programming practice to list the new tape before discarding the old, ensuring that the new tape is correct.

effort and no use of computer time. To do this, turn the LINE-OFF-LOCAL knob to LOCAL, depress the OFF button, correctly position the paper tape, and flick the switch to START. This procedure causes the tape to be printed at the console, independent of the computer.

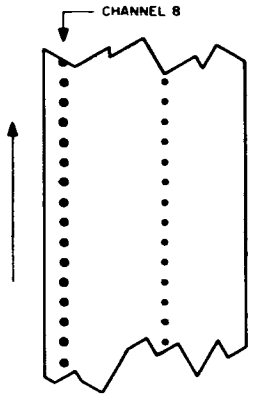
D.3 PAPER TAPE FORMATS

Data are recorded (punched) on paper tape by groups of holes arranged in a definite format along the length of the tape. The tape is divided into channels which run the length of the tape, and into columns which extend across the width of the tape as shown in the adjacent diagram. The paper-tape readers and punches used with the PDP-8/I computers accept 8-channel paper tape. The various formats are briefly explained and identified below.



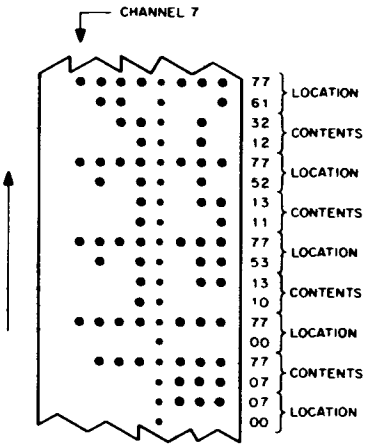
D.3.1 Leader/Trailer Format

Leader/trailer tape is used to introduce and conclude the object program when punched on paper tape. Leader/trailer tape can be recognized by a consistent channel 8 punch only as shown in the adjacent diagram.



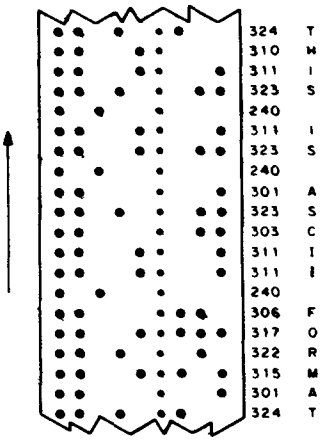
D.3.2 RIM Format

Paper tape punched in RIM format can be identified by the absence of a channel 8 punch, and by a channel 7 punch in every fourth column. The channel 7 punch indicates the start of a line of coding, and that (the first) column and the second column contain the location and the third and fourth columns contain the contents of the location.



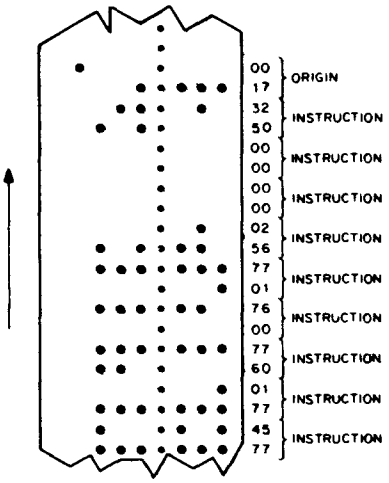
D.3.3 USASCII Format

USASCII (USA Standard Code for Information Interchange) format uses all eight channels to represent a single character (letter, number, or symbol) as shown in the adjacent diagram.



D.3.4 Binary Format

Binary format can be recognized by the absence of a channel 8 punch, an occasional channel 7 punch, and frequent sections of blank tape. The channel 7 punch denotes an origin of a program or subprogram or a change in origin, and subsequent columns contain the instructions (two columns per instruction) or data of succeeding locations.



APPENDIX E

SYSTEM CONFIGURATION AND OPTIONAL HARDWARE

E.1 INTRODUCTION

Depending on the hardware configuration of a particular TSS/8, there can be as many as 24 users working on the system simultaneously. The minimum hardware configuration is designed to service from four to eight users; it requires a 12K core memory (three fields). The first 8K (fields 0 and 1) are shared by the various Monitor subprograms. The other 4K (field 2) and any additional fields are shared by users of the system.

Each user has the following resources: 4K core memory for execution of programs and a corresponding 4K disk track for temporary storage of his core image when it is swapped out by the Monitor. The disk is divided as follows:

a. Monitor Area -- The first 20K of the disk is occupied by Monitor subprograms. These subprograms are swapped into core memory as needed.

b. User Swapping Area -- This area consists of a 4K track for each user in the system. When a user is temporarily swapped out of core, his program is stored on his 4K disk track.

c. File Storage Area -- The remaining disk area is used for storing System Library Programs and user files.

The system can have a maximum of 32K of core memory. Additional fields of core permit overlapping the run time of one user program with the swapping time of another, thus increasing operating speed.

A high-speed, paper-tape reader and punch unit is pictured in Figure E-1 and descriptions of the reader and punch units follow.

E.2 READER UNIT

The high-speed, paper-tape reader is used to photoelectrically input data into core memory from eight-channel fan-folded (non-oiled) punched paper tape at 300 characters per second. Power is applied to the reader when the computer POWER switch is turned on. The high-speed reader is under program control. However, tape can be advanced past the photoelectric sensors without causing input by pressing the FEED button (the white rectangular button shown in Figure E-1).

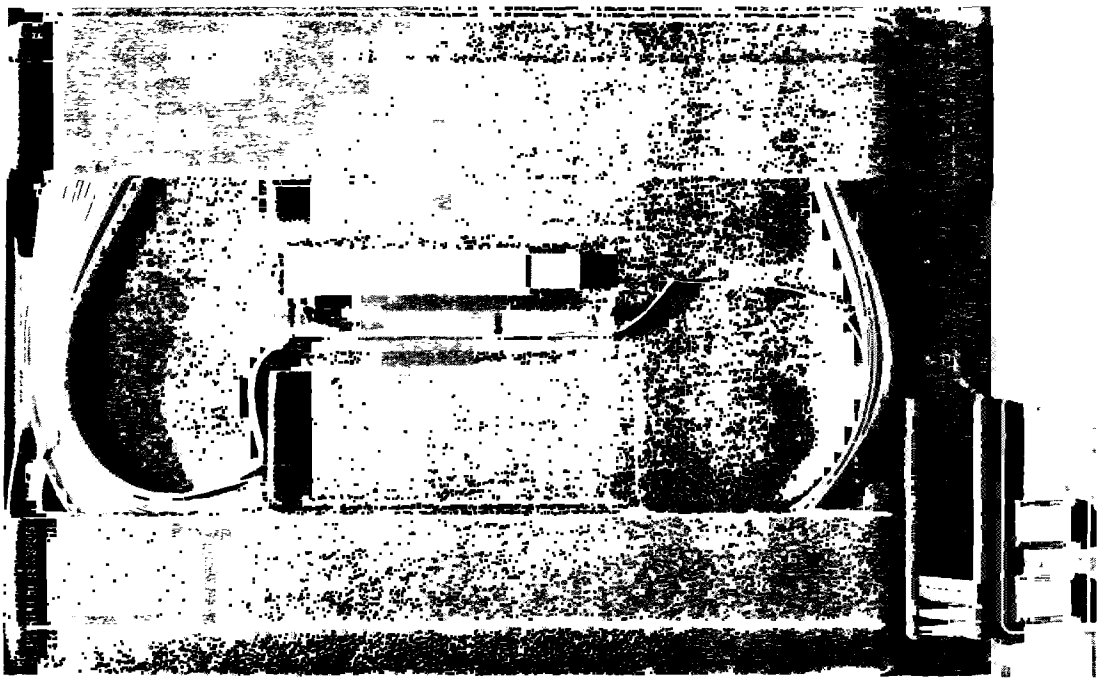


Figure E-1 High-Speed, Paper-Tape Reader and Punch Units

E.3 LOADING THE READER UNIT

The following is a description of the steps necessary to read a paper tape with the high-speed reader.

Step	Procedure
1	Raise tape retainer cover (located beneath the tape feed button).
2	Put tape into right-hand bin with channel one (see Section D.2) of the tape toward the rear of the bin.
3	Place several folds of leader tape through the reader and into the left-hand bin.
4	Place the tape over the reader head with feed holes engaged in the teeth of the sprocket wheel.
5	Close the tape retainer cover.
6	Depress the tape feed button (white rectangular button shown in Figure E-1) until leader tape is over the reader head.

CAUTION

Do not use oiled paper tape in the high-speed reader because oil collects dust and dirt which can cause reader errors.

E.4 PUNCH UNIT

The high-speed, paper-tape punch is used to record computer output on eight-channel fan-fold tape at 50 characters per second. All characters are punched under program control from the computer. Blank tape (feed holes only, no data) may be produced by pressing the FEED button (see Figure E-1). Power is applied to the punch when the POWER button on the punch unit is depressed (the punch motor can be heard). The two labeled buttons on the punch unit are described below.

POWER	This button is depressed to turn the punch ON and OFF.
FEED	While this button is depressed, the punch produces feed-hole-only punched tape for leader/trailer purposes.

E.5 DECTAPE CONTROL AND TRANSPORT UNITS

DECtape is a fast, convenient, input/output and data storage facility. The standard DECtape Transport unit is shown in Figure E-2.

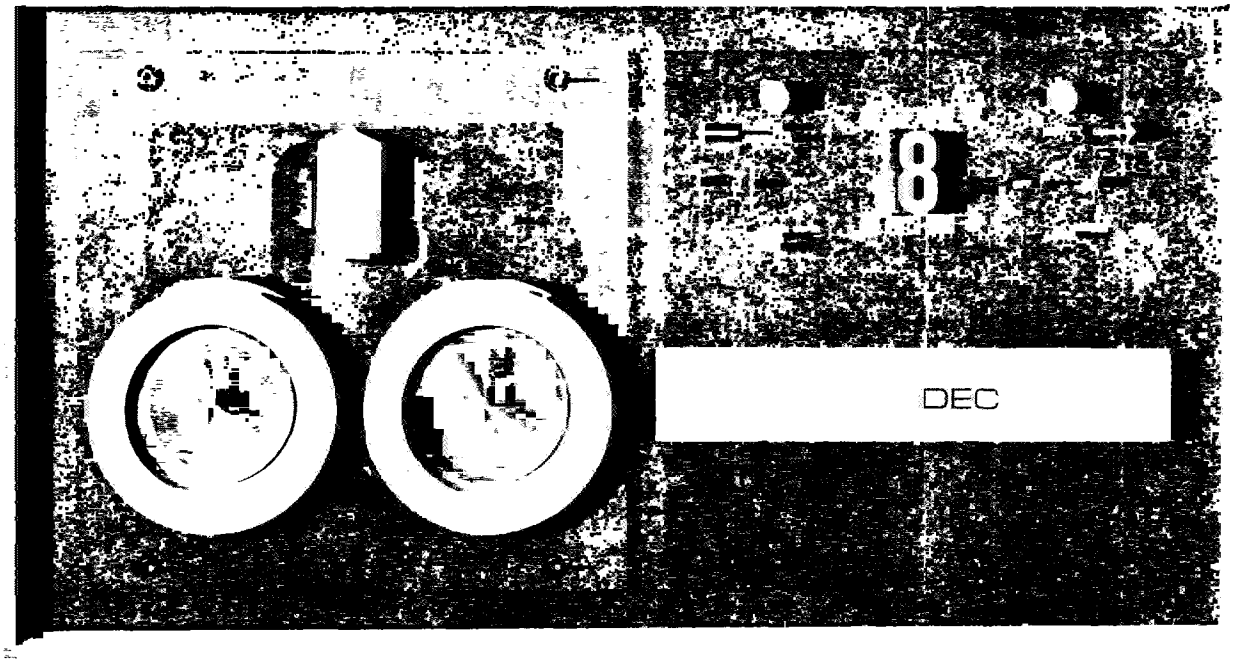


Figure E-2 DECtape Transport Unit

E.6 CONTROL UNIT

The DECtape control unit (located internally) interprets and controls the transfer of data between the computer and the transport unit. The DECtape control unit is located inside the rack containing the DECtape transport and can control up to eight separate DECtape transports.

E.7 TRANSPORT UNIT

The DECtape transport unit is a bidirectional magnetic tape transport utilizing a 10-track recording head to read and write five duplexed channels. Tape movement can be controlled by commands from the computer program or by the manual operation of switches located on the front panel of the transport; however, manual operation does not transfer data to the computer.

NOTE

Only certified DECtapes (pre-recorded with timing and marking tracks) should be used. Otherwise, the blank tape must be certified using the DECTOG Program (DEC-08-EUFA-D).

E.7.1 Transport Controls

The following is a description of the settings on the DECtape transport and their functions.

Settings	Function
➡	When depressed (must be in LOCAL mode), tape feeds onto right spool.
REMOTE	Transport is energized and under program control.
OFF	Transport is de-energized.
LOCAL	Transport is energized and under user control from external transport switches.
Unit Selector	Identifies the transport to the control unit.
WRITE ENABLED	DECtape is available for search, and write activities.
WRITE LOCK	DECtape is available for search and read activities only.
⬅	When depressed (must be in LOCAL mode), tape feeds onto left spool.

NOTE

The REMOTE and WRITE ENABLED lamps light to indicate the status of the transport.

E.7.2 Operating Procedure

The following is a description of how to mount a DECtape on a transport unit.

Step	Procedure
1	Set switch to OFF.
2	Place DECtape on left spindle with DECtape label out.
3	Wind four turns of tape on right spool.
4	Set switch to LOCAL.
5	Wind a few turns on right spindle with ➡ switch to make sure tape is properly mounted.
6	Dial correct unit number on unit selector (number 8 is equivalent to 0).
7	Set switch to REMOTE. Select either WRITE ENABLE or WRITE LOCK setting.

APPENDIX F
STORAGE ALLOCATION

F.1 STORAGE MAP

The system's storage allocation is illustrated below.

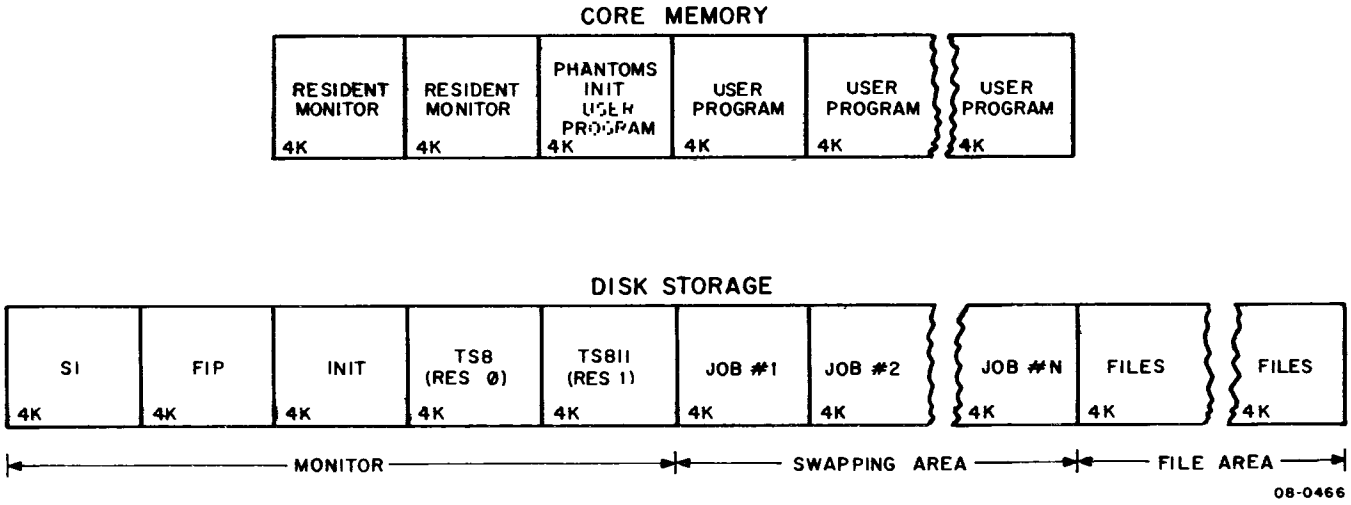


Figure F-1 TSS/8 Storage Map

F.2 FILE DIRECTORIES

There are two directories on the disk: the Master File Directory (MFD) referenced mainly by the system, and the User File Directory (UFD), referenced by the TSS/8 user. One of the functions of the MFD is to service the UFD. A UFD is a particular user's file directory containing the names of programs he has created on the disk.

The UFD is a file like any other file except that its filename is the project-programmer number and password. (See Section F.3 for an explanation.) When a user is logged in under a specific number and references the disk, he is actually referencing his own file area on the disk through the UFD which has his project-programmer number as its name. He can specifically code his routine to reference UFDs of other users or the MFD; whether he is successful or not depends on the type of protection that has been specified for the area he is trying to reference.

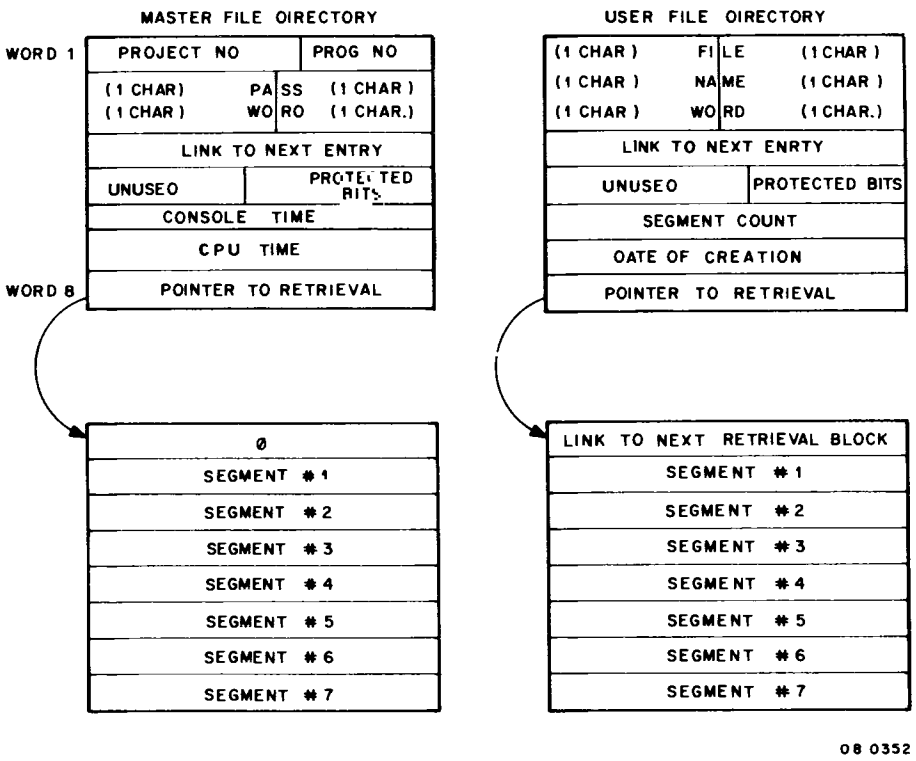


Figure F-2 File Directories

F.3 PROJECT-PROGRAMMER NUMBERS

System account numbers are a combination of project number and programmer number. The account number (always written in octal) has a binary equivalent. When expressed as a 12-bit binary number, the project number is formed by the leftmost 7 bits and the programmer number is the rightmost 5 bits. For example, the account number 623 (octal) equals 000 110 010 011 (binary). The left 7 bits (leading zeros included if any) are 0001100 (binary) and equal 14 (octal) which is the project number. The programmer number is 10011 (binary) or 23 (octal). Therefore a user with "account number" 623 has:

project number 14

programmer number 23

APPENDIX G

GLOSSARY OF ABBREVIATIONS AND TERMS

G.1 COMMONLY USED ABBREVIATIONS

The abbreviations listed below are used throughout the guide.

Abb.	Meaning	Abb.	Meaning
AC	Accumulator	HSP	High-Speed Punch
ADDR	Address	HSR	High-Speed Reader
B. SP.	Back Space	IF	Instruction Field
BIN	Binary	INST	Instruction
CLC	Current Location Counter	KBRD	Keyboard
		L	Link
CONT	Continue	LF	Line Feed
CR	Carriage Return	LOAD ADD	Load Address
CR/LF	Carriage Return-Line Feed	LOC	Location
		LSP	Low-Speed Punch
CTRL/FORM	Control/Form (which represents holding down the CTRL key while depressing the Form key).	LSR	Low-Speed Reader
		MA	Memory Address
		MB	Memory Buffer
		MQ	Multiplier Quotient
DEC	Digital Equipment Corporation	MRI	Memory Reference Instruction
		PC	Program Counter
DEP	Deposit	PROG	Program
DF	Data Field	REL	Release
EAE	Extended Arithmetic Element	RIM	Read-In Mode
		SING INST	Single Instruction
EXAM	Examine	SING STEP	Single Step
		SR	Switch Register
		TTY	Teletype

G.2 GLOSSARY OF TERMS

The following list of computer/programming terms is by no means complete. However, it does include many of the terms used in data processing.

Absolute Address (1) An address that is permanently assigned by the machine designer to a storage location.
(2) A pattern of characters that identifies a unique storage location without further modification.

Accumulator	A register in which the result of an operation is formed; Abbreviation AC
Acronym	A word formed from the first letter or letters of the successive words of a multiple word term.
Accuracy	The degree of freedom from error, i.e., the degree of conformity to truth or to a rule.
Address	A label, name, or number which designates a register or a location where information is stored. That part of an instruction which specifies the location of an operand.
Address Register	A register in which an address is stored.
Algorithm	A prescribed set of well-defined rules or processes for the solution of a problem in a finite number of steps.
Alphabet	An ordered set of unique representations called characters, e.g., the 26 letters of the Roman alphabet.
Alphanumeric	Pertaining to a character set that contains both letters and numerals, and usually other characters.
Arithmetic Unit	The component of a computer where arithmetic and logical operations are performed.
ASCII	An abbreviation for USA Standard Code for Information Interchange.
Assemble	To translate from a symbolic (source) program to a machine language (object) program by substituting binary operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.
Assembler	A program that assembles.
Auto-Indexing	When an absolute location 0010 through 0017 is addressed indirectly, the content of that location is incremented by one, rewritten in that same location, and then read as the effective address of the next instruction.
Auxiliary Operation	An operation performed by equipment not under direct control of the computer. Off-line operation.
Auxiliary Storage	Storage that supplements the primary storage.
Binary	(1) Pertaining to a characteristic or property involving a selection, choice, or condition in which there are two possibilities. (2) Pertaining to the numeration system with a radix of two.
Binary Digit	One of the symbols 1 or 0. A digit in the binary scale of notation; called a bit.
Bit	A binary digit.
Blank Character	A character used to produce a space on an output device.
Block	A set of things, such as words, characters, or digits, handled as a unit.

Bootstrap	A technique or device designed to bring itself into a desired state by means of its own action, e.g., a routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.
Branch	A point in a routine where one of two or more choices is made under control of the routine, i.e., a conditional transfer (jump).
Buffer Storage	A part of core memory where information is stored temporarily during transfer; it may attempt to match the speeds of internal computation and the I/O device, thus permitting simultaneous computation and input/output.
Byte	A group of binary digits usually operated upon as a unit, e.g., 8-bit or b-bit byte.
Call	To transfer control to a specified routine.
Calling Sequence	A specified set of instructions and data necessary to set up and call a given routine.
Carriage Return	The Teletype operation that causes the next character to be printed at the left margin.
Central Processing Unit	The unit of a computing system that includes the circuits controlling the interpretation and execution of instruction; the computer proper, excluding I/O and other peripheral devices.
Character	A single letter, numeral, or space mark used to represent information.
Clear	To erase the contents of a storage location by replacing the contents with blanks or zeros.
Closed Subroutine	A subroutine not stored in the main part of a program. Such a subroutine is entered by a jump operation and provision is made to return control to the main routine at the end of the surbroutine.
Coding	To write instructions for a computer using symbols meaningful to the computer.
Command	A control signal, usually written as a character or group of characters, to direct action by a system program.
Compile	To produce a machine language routine from a routine written in source language by selecting appropriate subroutines from a subroutine library, as directed by the instructions or other symbols of the original routine, supplying the linkage which combines the subroutines into a workable routine and translating the subroutines and linkage into machine language.
Compiler	A program that compiles.
Complement	To form the negative of a binary word by replacing all 0 bits with 1 bits and vice versa.
Computer	A device capable of accepting information, processing it, and providing the results in a usable form.
Computer Program	A plan or routine for solving a problem on a computer.

Computer Word	A sequence of 12 bits treated as a unit and capable of being stored in one computer location.
Console	Usually the external front side of a device where controls and indicators are available for manual operation of the device.
Control Character	A character whose occurrence in a particular context initiates, modifies, or stops a control operation, e.g., a character to control carriage return.
Control Panel	The part of a device console that contains manual controls.
Convert	To change the representation of data from one form to another.
Copy	To reproduce data, leaving the original data unchanged.
Core Memory	The main storage device in the PDP-8 in which binary data is represented by the direction of magnetization in each unit of an array of magnetic material.
Cycle	To repeat a set of operations until a stated condition is met.
Cycle Time	An interval of time in which one set of events is completed.
Data	A general term used to denote any or all facts, numbers, letters, and symbols. It connotes basic elements of information which can be processed or produced by a computer.
Data Break	A facility which permits I/O transfers to occur simultaneously with program execution on a cycle-stealing basis.
Debug	To detect, locate, and correct mistakes in a program.
Decision	A determination of future action.
Delay	The amount of time by which an event is retarded.
Delimiter	A character that separates and organizes items of data.
Diagnostic	Pertaining to the detection and isolation of a malfunction or mistake.
Digit	A character used to represent one of the non-negative integers smaller than the radix, e.g., in binary notation, either 0 or 1.
Digital Computer	A device that operates on discrete data, performing sequences of arithmetic and logical operations on this data.
Direct Address	An address that specifies the location of an operand.
Display	A visual presentation of data.
Document	A medium on which information is recorded for human or machine use.
Double Precision	Pertaining to the use of two computer words to represent a number.
Downtime	The time interval during which a device is inoperative.

Dummy	An artificial address, instruction, or record of information inserted solely to fulfill prescribed conditions.
Dump	To copy the contents of all or part of core memory, usually onto an external storage medium.
Dynamic Dump	A dump that is performed during the execution of a program.
Edit	To rearrange information for machine input or output.
Effective Address	The address actually used in a particular execution of a computer instruction.
End-Around Carry	The action of adding the most significant bit of a binary number to the least significant bit.
Execute	To carry out an instruction or run a program on the computer.
Executive Routine	A routine that controls or monitors the execution of other routines.
External Storage	A facility or device, not an integral part of the computer, on which data usable by the computer is stored, such as paper tape, DECtape, or DECdisk.
File	A collection of related records treated as a unit, generally data or a program.
Fixed Point	In a numeration system the position of the radix point is fixed with respect to one end of the numerals, according to some convention.
Flip-Flop	A basic computer circuit or device capable of assuming either one of two stable states at a given time.
Floating Point	A numeration system in which the position of the radix point is indicated by one part (the exponent part), the other part represents the significant digits (the fractional part).
Flowchart	A graphical representation of the sequence of instructions required to carry out a data processing operation.
Format	The arrangement of data.
Function	A specific purpose of an entity or its characteristic action.
Hardware	Physical equipment, e.g., mechanical, electrical, or electronic devices.
Head	A device that reads, records, or erases data on a storage device.
Heuristic	Pertaining to exploratory methods of problem solving.
I/O	Input or output or both.
Identifier	A symbol whose purpose is to identify, indicate, or name a body of data.
Indirect Address	An address in a computer instruction which indicates a location where the address of the referenced operand is to be found.

Initialize	To set counters, switches, and addresses to zero or other starting values at the beginning of, or at prescribed points in, a computer routine.
Input	The transferring of data from auxiliary or external storage into the internal storage of the computer.
Instruction	A set of bits (in an object program) or characters (in a source program) which as a unit cause the computer to perform an operation.
Internal Storage	The storage facilities forming an integral physical part of the computer and directly controlled by the computer. Also called main memory and core memory.
Interrupt	To stop a process in such a way that it can be resumed.
Jump	A departure from the normal sequence of executing instructions in a computer.
Label	An identifier.
Language	A set of representations, conventions, and rules used to convey information.
Leader	The blank section of tape at the beginning of the tape.
Least Significant Digit	The rightmost digit of a binary number.
Library	An organized collection of standard and proven routines and subroutines which can be incorporated in larger routines.
Library Routine	A proven routine that is maintained in a program library.
Load	To place data into internal storage.
Location	A place in storage or memory where a unit of data or an instruction may be stored.
Loop	A sequence of instructions that is executed repeatedly until a terminal condition prevails.
Machine Instruction	An instruction written in machine language.
Machine Language	A language designed for interpretation and use by the machine without translation.
Macro Instruction	An instruction in a source language that is equivalent to a specified sequence of machine instructions.
Manual Input	The entry of data by hand into a device at the time of processing.
Manual Operation	The processing of data in a system by direct manual techniques.
Memory	(1) The erasable storage in the computer. (2) Pertaining to a device in which data can be stored and from which it can be retrieved.

No Op	An instruction that specifically instructs the computer to do nothing, except to proceed to the next instruction in sequence.
Object Program	The machine language program which is the output after translation from the source language. The binary program which runs on the computer.
Octal	(1) Pertaining to a characteristic or property involving a selection, choice, or condition in which there are eight possibilities. (2) Pertaining to the numeration system with a radix of eight.
Off Line	Pertaining to equipment or devices not under direct control of the computer.
On Line	Pertaining to equipment or devices under direct control of the computer; also to programs operating directly and immediately to user commands, e.g., FOCAL and DDT.
Open Subroutine	A subroutine that must be relocated and inserted into a routine at each place it is used.
Operand	That which is effected, manipulated, or operated upon.
Origin	The absolute address of the beginning of a program.
Output	Information transferred from the internal storage of a computer to output devices or external storage.
Overflow	The generation of a quantity beyond the capacity of a register.
Page	In the PDP-8/I, a unit of 200 (octal) locations which may be addressed directly.
Patch	To modify a routine in a rough or expedient way.
Predefined Process	A named process consisting of one or more operations or program steps that are specified elsewhere in a routine.
Procedure	The course of action taken for the solution of a problem.
Processor	A computer program that includes the compiling, assembling, translating, and related functions for a specific programming language.
Program	The complete sequence of instructions and routines necessary to solve a problem.
Program Library	A collection of available computer programs and routines.
Programming Language	A language used to prepare computer programs.
Protected Location	A storage location reserved for special purposes in which data cannot be stored without undergoing a screening procedure to establish suitability for storage therein.
Punched Paper Tape	A paper tape on which a pattern of holes is used to represent data.
Pushdown List	A list that is constructed and maintained so that the next item to be retrieved is the most recently stored item in the list, i.e., last in, first out.

Radix	The quantity of characters for use in each of the digital positions of a numbering system.
Read	To transfer information from an input device to internal storage; also refers to the internal acquisition of data from memory.
Real Time	Pertaining to computation performed while the related physical process is taking place so that results of the computation can be used in guiding the physical process.
Record	A collection of related items of data, treated as a unit.
Register	A device capable of storing a specified amount of data, such as one word.
Reset	To restore a storage device to a prescribed state.
Restart	To re-establish the execution of a program.
Routine	A set of instructions arranged in proper sequence to cause the computer to perform a desired task.
Run	A single, continuous performance of a program.
Scan	To examine sequentially part by part.
Search	To examine a set of items for those that have a desired property.
Set	To place a storage device into a specified state.
Single Step	Operation of the computer in which each instruction is performed in response to a single manual operation.
Skip	To ignore one or more instructions in a sequence of instructions.
Software	The collection of programs and routines associated with the computer.
Source Language	A symbolic language that is an input to a given translation process.
Source Program	A program written in a symbolic (source) language.
Statement	A meaningful expression or generalized instruction in a source language.
Step	One operation in a routine.
Storage Allocation	The assignment of blocks of data to specified blocks of storage.
Storage Capacity	The amount of data that can be contained in a storage device.
Storage Device	A device into which data can be entered, in which it can be held, and from which it can be retrieved.
Store	To enter data into a storage device.

String	A connected sequence of entities such as characters in a command string.
Subroutine	A routine that can be part of another routine.
Switch	A device or programming technique for making selections.
Symbolic Address	An address expressed in symbols convenient to the programmer. A label.
Symbolic Coding	Writing instructions using symbolic notation instead of actual machine instruction notation.
System	An assembly of software and hardware united to form an organized whole.
Tape Drive	A device that moves tape past a head.
Temporary Storage	Storage locations reserved for intermediate results.
Terminal	A point in a system at which data can either enter or leave.
Time Sharing	The interleaving of the time of a device.
Toggle	Pertaining to the operation of a flip-flop or switch.
Translate	To convert from one language to another.
Underflow	The condition that arises when a computation yields a result whose magnitude is smaller than the system is capable of representing.
Variable	A quantity that can assume any of a given set of values.
Word	A 12-bit unit of data in the PDP-8/I which may be stored in one addressable location.
Word Length	The number of bits in a word.
Write	To transfer information from internal storage to an output device or to auxiliary storage.

INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

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

Digital Software News for the PDP-8 Family
Digital Software News for the PDP-9/15 Family

These newsletters contain information to update the cumulative

Software Performance Summary for the PDP-8 Family
Software Performance Summary for the PDP-9/15 Family

The appropriate edition of the Software Performance Summary is included in each basic software kit for new customers. Additional copies may be requested without charge.

Any questions or problems on the articles contained in these publications or concerning the use of Digital's software should be reported to the Software Specialist or Sales Engineer at the nearest Digital office.

New and revised software and manuals, current issues of the Software Performance Summary, and cumulative Software Manual Updates are available from the Program Library. To place an order, please contact your local Digital office or write to:

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

When ordering, include the code number and a brief description of the program or manual requested.

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

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

Please complete the return postcard below if you would like to receive Digital's newsletters

Send ☐ Digital Software News for the PDP-8 Family, or
☐ Digital Software News for the PDP-9/15 Family

To Name_____
Company Name _____
Address _____
_____
_____
(zip code)

My computer is a ☐ PDP-8I ☐ PDP-12 ☐ PDP-9
☐ PDP-8L ☐ LINC-8 ☐ PDP-15
☐ PDP-8S ☐ Other _____

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

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

READER'S COMMENTS

TSS/8 TIME-SHARING
SYSTEM USER'S GUIDE
DEC-T8-MRFB-D

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback – your critical evaluation of this manual.

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

Did you find errors in this manual? Yes: Page 4-8, Error Code Explanation Table - number "Floating-point number larger than 2047".

How can this manual be improved?

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

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

Please describe your position. _____

Name _____ Organization _____

Street _____ Department _____

City _____ State _____ Zip or Country _____

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by

digital

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