August 1978

The TRAX Support Environment User's Guide is a combination tutorial/reference manual directed to programmers with a wide range of technical background.

TRAX
Support Environment
User's Guide

AA-D331A-TC

OPERATING SYSTEM AND VERSION: TRAX Version 1.0

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard. massachusetts

First Printing, August 1978

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10

CONTENTS

CONTENTS (CONT.)

CONTENTS (CONT.)

CONTENTS (CONT.)

PREFACE

This manual has two main divisions. Part I is primarily tutorial. You are assumed to have some experience with programming and interactive terminal operation, but little or no prior experience with the TRAX Support Environment. Part I consists of seven chapters:

- Chapter 1 explains the purpose and design philosophy of TRAX and the TRAX Support Environment, including the Digital Command Language (DCL).
- Chapter 2 introduces interactive terminal operation in the TRAX Support Environment, using a simple annotated terminal session.
- Chapter 3 introduces file creation and management, including the format of the file specification used to identify files.
- Chapter 4 describes basic device handling in the TRAX Support Environment.
- Chapter 5 describes the process of developing programs into executable tasks.
- Chapter 6 explains the fundamentals of batch processing.
- Chapter 7 explains the use of indirect command files.

Part II consists of reference information. It covers in detail many subjects covered only generally in Part I.

- Chapter 8 explains command syntax.
- Chapter 9 describes the set of TRAX commands available to the general user. Commands are presented in alphabetical order.

The TRAX Support Environment described herein provides a traditional command language environment. It is designed to assist you in developing transaction step tasks and tasks that augment the transaction processing system such as reports. Through it, facilities, such as source language compilers and assemblers the DEC EDITOR, and the Linker.

This manual does not describe these other software facilities in detail, however. Rather, it explains in general what they are, what they do, and how to invoke or access them. To use them effectively, you will need to consult other manuals in the TRAX documentation set. These manuals include:

1. For language and compiler interface information:

   TRAX BASIC-PLUS-2 Language Reference Manual (Order No. AA-D366A-TC)
   TRAX BASIC-PLUS-2 Language User's Guide (Order No. AA-D377A-TC)
   TRAX COBOL Language Reference Manual (Order No. AA-D338A-TC)
   TRAX COBOL Language User's Guide (Order No. AA-D339A-TC)
   TRAX MACRO Language Reference Manual (Order No. AA-D340A-TC)
   TRAX RMS MACRO Programmer's Guide (Order No. AA-D344A-TC)

2. For file creation and manipulation:

   DEC EDITOR Reference Manual (Order No. AA-D347A-TC)
   TRAX SORT Reference Manual (Order No. AA-D346A-TC)

3. For linking:

   TRAX Linker Reference Manual (Order No. AA-D342A-TC)
   TRAX System Manager's Guide (Order No. AA-D332A-TC)

PART ONE

USING THE TRAX SUPPORT ENVIRONMENT

– A TUTORIAL –

CHAPTER 1

INTRODUCTION TO TRAX SUPPORT ENVIRONMENT

1.1 WHAT IS TRAX?

One can regard TRAX as two distinct but complementary operating environments: the Transaction Processing Environment and the Support Environment.

The transaction processing environment is oriented to the operation of an established transaction-processing application by an individual at an application terminal. Such an individual knows the interactions of the application, but is usually not a programmer.

The Support Environment is intended for program development, for control and monitoring of transaction processing, and for subsidiary applications that do not use transaction processing functions, such as the running of routine inventory reports or customer billing. It is a traditional command-language facility by which you can develop and run stand alone support programs, either interactively or through batch processing.

1.2 THE SUPPORT ENVIRONMENT

As a Support Environment user, you will work by entering commands at a terminal and control only the terminal at which you log in. You will issue commands using a version of DIGITAL Command Language (DCL) designed specially for TRAX.

The Support Environment is used for the following types of processing:

- Controlling and supervising of transaction processing, batch processing, and spooling
- Running of support programs related to transaction applications
- Editing of source program and data files
- Compiling, linking, and debugging of programs
- Defining and implementing transaction processors
- File backup and recovery

Programs running in the Support Environment are called support programs, and a TRAX terminal logged into the Support Environment is called a support terminal. Such programs and terminals cannot modify files to which a running transaction processor has write access. Support programs run under control of a support terminal or a batch processor.

The basic unit of executable code is called a task. Each DCL command that you enter and each program that you run is a task.

To qualify as a task, a program must be compiled and linked with all necessary support routines. Linking a program forms an executable task image and stores it as a permanent file on disk. When you command the system to run a task, the system retrieves the executable task image from the file that you specify.

The system schedules the running of tasks according to the priority of the task and the availability of system resources. The resources include computer memory and devices needed to perform input/output. An efficient scheduling technique allows many tasks to be processed simultaneously on a demand basis.

Every Support Environment user has a unique identification, and each terminal has a unique device name. These identifiers key the system as to the origin of each command, each task, and each data input, and also direct output to the proper destination.

The Support Environment offers two general methods of command processing:

- Interactive command processing
- Batch command processing

In either method, you direct the system by means of commands. In interactive processing, you type commands one at a time in response to a prompt from the system. In batch processing, you create a file containing commands for each operation you want performed, together with all data that you expect the system to request while processing the commands. Batch jobs are processed on virtual terminals created by the system; this frees your terminal for other activity.

You can reference indirect command files during interactive sessions or batch processing jobs. An indirect command file is a file consisting of one or more interactive commands to be executed as a unit. If you have a sequence of commands that you use fairly often or a long, complex command, you can create a file containing this command information. Later you can invoke this indirect command file by a simple command.

The DCL command language provides the following general capabilities:

1. Beginning and ending an interactive session or batch job
2. Creating, editing, and managing files
3. Allocating and controlling devices
4. Developing and running programs
5. Monitoring and controlling program execution

1.3 INTERACTIVE COMMAND PROCESSING

All communication between user and system occurs during a terminal session. The user initiates a terminal session by logging in and terminates a session by logging out.

Interactive command processing is conversational in nature. It consists of a two-way communication between you and the system. You initiate each action with a command, entering commands one at a time. After entering a command, you wait for the system to perform the requested action. When the system completes processing of your command or determines that it cannot comply, it informs you accordingly with prompts or error messages.

1.3.1 Prompts

Various prompts inform you as to when the system expects input and what type of input it expects. The prompt character (>) indicates that a DCL command is expected.

If you do not supply all the information necessary to execute a command, the system will prompt for required items. The COPY command, for example, requires the name of an input file (the file to be copied) and an output file (the copy). If you do not specify these files, TRAX will prompt for them.

```
>COPY
FROM? NEWFILE.DAT
TO? NEWFILE2.DAT
```

Some commands invoke system functions that have prompting modes of their own. EDIT is one such command.

The following example shows three different types of prompts:

```
>EDIT
FILE? MYPROG.CBL
*
```

As noted previously, the > symbol is the DCL prompt; this means that whatever is entered next will be treated as a DCL command. In this example you respond to the prompt by typing an EDIT command to invoke the DEC EDITOR.

EDIT is one of many DCL commands that can prompt for command parameters. The editor program needs a file, and the system prompts for a file specification by typing FILE?.

After you specify the file to be edited, the system locates the file and makes it available to the editor. At this point, the editor is ready to process editing commands and signifies this by displaying its own prompt–an asterisk.

1.3.2 Error Messages

An error message can occur for various reasons, and the contents of the message usually gives an indication of the problem. For example:

```
>EDIT ABCXYZ
EDI -- FILENAME OR FILETYPE NOT SPECIFIED
>
>
```

The message occurs because required information was omitted. (The EDIT command requires that you specify both the file name and file type components of the file you want to edit.) On the next line, the system prints the DCL prompt character (>) allowing you to enter the command correctly.

1.4 BATCH FILE PROCESSING

Batch processing allows you to execute a terminal session off-line. The batch processor creates a virtual terminal for the batch job. A virtual terminal is an entity with the logical attributes of an interactive terminal. By creating a virtual terminal as a processing medium for your batch job, the system frees your interactive terminal for other use.

Instead of entering commands one at a time and entering data interactively, you create a file containing all information that you would enter during an interactive session. This information includes commands and any data that you expect the tasks to require during execution. This file, or series of files, is submitted to the batch processor and is called a batch job.

For the most part, command lines in a batch file are the same as command lines entered interactively, but there are differences:

1. Command lines in a batch file must be identified as such by a dollar sign ($) as the first character of the command line. Lines that do not begin with a dollar sign are treated as data for the preceding command.
2. Every batch command file begins with a $JOB command and ends with an $EOJ command. These are logically parallel to the LOGIN and LOGOUT commands that begin and end an interactive session.
3. The batch command set includes commands that can specify alternative actions, responsive to processing conditions. This allows monitoring of job execution. Batch commands can be labeled to facilitate skipping of commands.

Each batch job produces a log file that records its activity. When listed, it provides a hard copy record of the job similar to information that appears on the terminal during an interactive session.

See Chapter 6 for a detailed description of batch processing.

1.5 INDIRECT COMMAND FILE PROCESSING

An indirect command file is a file containing a fixed sequence of commands. Unlike a batch file, it cannot contain data, and commands must be in interactive format (no dollar sign prefix, no labels, and no conditionals). It does not constitute a terminal session, only an adjunct to the terminal session that calls it.

Certain specific command sequences occur fairly often. Also, some individual commands can be rather long and complicated. You can create indirect command files containing such command sequences.

To execute an indirect command file, type an at sign (@) followed by the file specification of the indirect file. The system then retrieves the indirect command file and executes the commands contained therein as though they had been entered directly through the terminal keyboard or included in the batch stream.

One key difference between batch and indirect file processing is in the way data is supplied to tasks. Tasks initiated by batch jobs obtain input from data blocks in the batch stream, while tasks initiated by commands in an indirect file expect interactive input, entered at the issuing terminal.

Moreover, an indirect command file is executed immediately, on the same terminal. A batch file requires the system to create a virtual terminal. Its time of execution is uncertain, because it competes with other batch jobs for processing by a batch processor.

See Chapter 7 for a detailed description of indirect command file use.

1.6 PROGRAMMING LANGUAGES

Two programming languages–BASIC-PLUS-2 AND COBOL–are supportable in the Support Environment. Each compiler is called by a DCL command that you can use in both interactive and batch mode, or can include in indirect files. Chapter 5 explains how these languages are used in the Support Environment and how source programs are processed into executable tasks, using the TRAX Linker. For detailed information on a programming language, refer to the language reference manual and user's guide for the particular language.

1.6.1 BASIC-PLUS-2 Language

The BASIC language is easy to learn and is widely used in educational, business, and scientific applications. The BASIC compiler available to the Support Environment is called BASIC-PLUS-2 and includes many advanced features. In this manual the term BASIC generally refers to the programming language. The term BASIC-PLUS-2 is used only when necessary to emphasize attributes of the BASIC-PLUS-2 compiler.

1.6.2 COBOL

COBOL (Common Business Oriented Language) is a pseudo-English language designed primarily for business applications. The TRAX Support Environment uses the PDP-11 COBOL compiler. This compiler uses a terminal-oriented line format.

TRAX COBOL conforms to the American National Standard Programming Language COBOL, ANSIX3.23-1974, level 1, and offers many higher level features. Several utility programs are provided.

1.7 FILE STORAGE

The Support Environment provides a set of commands for storing and maintaining information. All information is stored in logical units called files.

A file is defined as an ordered collection of information. The maximum size of a file is one disk volume. A file can be empty and occupy no disk storage space. If a file contains information, its minimum size is one disk block (512 bytes).

As a Support Environment user, you will encounter may types of files: source program files, data files, compiled object files, command files, task images, and batch files, to name only a few of the most common types. The system provides a standard set of mnemonic file type identifiers that you can use by default, and also allows you to define file types.

Each file is identified by a unique file specification. The file specification contains several details: the storage device, the directory on which the file existence is recorded, the file name, the file type, and the version number.

Any file can be protected against unauthorized access by means of a file security facility.

Every user has at least one User File Directory (UFD). This is a special file that lists all the files belonging to the user. Chapter 3 describes file management techniques in detail, and Chapter 4 describes the techniques of handling the devices and volumes on which files are maintained.

CHAPTER 2

USING THE TRAX TERMINAL

This chapter introduces terminal operation in the TRAX Support Environment. It describes the terminal keyboard and illustrates, by means of a sample session at an LA36 terminal, how to issue various DCL commands.

2.1 THE KEYBOARD

The interactive user enters information into the system from a terminal. Various types of terminals can be connected to the system. All have a keyboard that is similar to the keyboard of a typewriter. Number and letter keys are in traditional typewriter format, but punctuation and special characters may be in different positions from one type of terminal to another. Also, terminals have special function keys that typewriters do not have; these vary from one terminal to another and in some cases may have different names. These functions are described in Chapter 8.

Figure 2-1 shows the keyboard layout for the LA36 and VT52 terminals. These terminals are supported at TRAX installations. The LA36 is a DECwriter terminal that uses a hard-copy (that is, character printing) display, and the VT52 uses a video display.

2.2 A SAMPLE INTERACTIVE SESSION

You communicate with the system by typing commands. Each command defines an action for the system to perform.

This section demonstrates the use of DCL commands in an interactive terminal session. Figure 2-2 shows the actual commands and system response.

NOTE

In this example, as in some other examples later in the manual, all system output is printed in red, while all input by the terminal operator is shown in black. The numbers in the left margin are for reference, and do not appear in the actual listing.

Every interactive session begins with a LOGIN command (1) to access the system. After you type the command name LOGIN, the system responds by displaying USERID?; you respond by typing your User Name, which in this case is SAMPLE. Then the system requests you to enter your password, a string known only to you and the system. Notice that the password string is not displayed as you type it.

You complete the LOGIN sequence, and the system responds with an acknowledging message. When the system displays the prompt (>), it is ready to receive command input.

The first word of every command line (that is, the first work you type after the prompt (>) appears) is a command name. As you will see later in this session, some commands require only

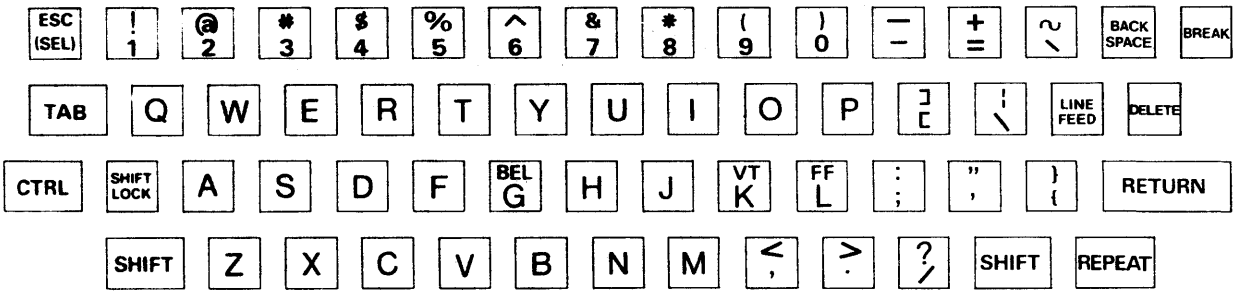

Figure 2-1 LA36/VT52 Keyboard Layout

```
(1) >LOGIN
    USERID? SAMPLE
    PASSWORD:

           TRAX VERSION 1.0A SYSTEM

    GOOD AFTERNOON
    18-JUL-78 14:37 LOGGED ON TERMINAL TT4:




(2) >DIRECTORY


    DIRECTORY DB0:[40,40]
    18-JUL-78 14:38

    A.LST;1              1.          18-JUL-78 14:14
    A.ODL;1              1.          18-JUL-78 14:14
    A.OBJ;1              2.          18-JUL-78 14:14
    A.TSK;1              27.      C  18-JUL-78 14:14
    A.CBL;2              1.          18-JUL-78 14:15

    TOTAL OF 32./32. BLOCKS IN 5. FILES


    >DIRECTORY TEST2

(3) DIR -- NO SUCH FILE(S)


(4) >EDIT TEST2.B2S
    *I
    100     PRINT 'THIS IS A SIMPLE BASIC PROGRAM.'
(5) 32767   END
    ^Z

    *EXIT
    2 LINES OUTPUT


(6) >TYPE TEST2.B2S
    100     PRINT 'THIS IS A SIMPLE BASIC PROGRAM.'
    32767   END
(7) >BASIC

    Basic Plus 2     V01-53


    Basic2
```

Figure 2-2 Sample Terminal Session

```
    OLD TEST2

    Basic2

    COMPILE TEST2

    Basic2

    BUILD TEST2

    Basic2
```

(8) EXIT

(9) >LINK/BASIC TEST2

(10) >RUN TEST2
THIS IS A SIMPLE BASIC PROGRAM.

(11) >DIRECTORY TEST2.*

```
DIRECTORY DB0:[40,40]
18-JUL-78 14:47

TEST2.B2S;1          1.          18-JUL-78 14:43
TEST2.OBJ;1          2.          18-JUL-78 14:45
TEST2.CMD;1          1.          18-JUL-78 14:45
TEST2.ODL;1          1.          18-JUL-78 14:45
TEST2.TSK;1          19.      C  18-JUL-78 14:46

TOTAL OF 24./32. BLOCKS IN 5. FILES
```

(12) >COPY
FROM? TEST2.TSK
TO? [40,41]
>

(13) >COPY [350,230]AMORT.*
TO? [40,40]

(14) >RENAME AMORT.*;* INTRST.*;*

(15) >TYPE AMORT.B2S
TYP -- NO SUCH FILE(S)
SY0:[40,40]AMORT.B2S

(16)
```
>TYPE INTRST.B2S
10      input 'interest' j
20      let j=j/100
30      input 'amount' a
40      input 'number of years';n
50      input 'payments per year'; m
60      let n=n*m \ i=j/m \ b=1+i
70      let r=a*i/(1-1/b^n)
100     print'amount per payment=';int(r*10^2+.5)/10^2
110     print 'total interest     =';int((r*n-a)*10^2+.5)/10^2
1000    end
```

(17)
```
>RUN INTRST
interest
? 10
amount
? 5000
number of years? 5
payments per year? 12
amount per payment= 106.24
total interest     = 1374.11
```

(18)
```
>DIRECTORY [*,*]

DIRECTORY DB0:[1,1]
19-JUL-78 11:43

OLDCOBLIB.OLB;1      217.        07-JUN-78 08:11
VMLIB.OLB;1          17.      C  19-MAY-78 10:21
COBRTS.CMD;11        1.          24-JUN-78 09:26
SYSLIB.OLB;2         183.        19-MAY-78 10:20
JUN20925.CDA;1       2561.       20-JUN-78 09:25
```

(19)
```
DCL>ABORT DIRECTORY
RMSLIB.OLB;
```

(20)
```
11:44:20    TASK "DIRT4 " TERMINATED
            ABORTED VIA DIRECTIVE OR MCR
            AND WITH PENDING IO REQUESTS
>
```

(21)
```
>LOGOUT
 TRAX
18-JUL-78 16:00 TT4:  LOGGED OFF
>
```

a command name to fully define their action, while other commands require parameters or qualifiers to be meaningful.

The DIRECTORY command is meaningful as a single-word command, although you can alter its function by using parameters to specify files and qualifiers to request different types of information about the files.

When you enter a DIRECTORY command with no parameter (2), you obtain a listing of all the files in your directory. For the present, think of a directory as a set of files belonging to a particular user. Notice the first of the DIRECTORY output; it indicates that your files are on the disk DBO=, and that your directory is designated [40,40].

You can also enter a DIRECTORY command specifying particular files. If you specify files that are not in the directory, you receive an error message (3). Every file specification has several components, as will be explained in Chapter 3; this particular file specification requests a list of all files in your directory with file name TEST2.

The asterisk is a wildcard in the file type component and indicates that all files named TEMP2 should be listed, regardless of file type.

In response, the system informs you that no such files exist at present. Thus you can use this file name without compromising other files.

Next you create a simple BASIC source program, using the DEC EDITOR. You invoke the DEC EDITOR by entering an EDIT command. The Editor cannot operate until you have indicated exactly which file you want to edit. If you do not specify a file when you type EDIT, the system prompts for a file specification by displaying the word, FILE?. Many DCL commands have similar prompts.

After you have typed the EDIT command (4), specifying TEST2. B2S as the file, an asterisk appears. This is the editor's prompt symbol, indicating that you are expected to enter an editor command. That is, you have entered editor command mode. The command I is an abbreviation for INPUT and alerts the file to include whatever you type next in the file.

After entering the BASIC source code (5), you signal the editor that you have completed the input information by typing CTRL/Z. You do this by holding down the CTRL key while typing Z. The * prompt reappears, and, because you are finished with the editor, you type EXIT. This ends editor command mode, and completes the creation of your file.

The TYPE command lists the file, allowing you to check its content (6).

To build the BASIC source file into an executable task, you enter the BASIC-PLUS-2 compiler facility by entering the BASIC command (7).

Under BASIC control, you use the commands of that facility to identify the source file, compile it into object code, and prepare the file for linking a command file. Then an EXIT command returns your terminal to DCL control (8).

NOTE

This manual describes the features of DEC EDITOR and the programming languages to the extent necessary to explain their access relationships with DCL. Before attempting to use any editing or programming language, you should study the appropriate reference manual or user's guide.

A program must be linked with the system before it can be executed; that is, before it constitutes a task. After linking this simple program (9), you can run it (10).

The string /BASIC appended to the command name LINK is a qualifier.

This particular qualifier informs the Linker program that the command file TEST2.CMD was produced by the BASIC-PLUS-2 facility and requires special processing. Either a command name or parameter can be qualified. Qualifiers always begin with a slash character (/) and are system keywords appended directly to the command name or parameter.

Now that the BASIC program has been processed, the command

```
DIRECTORY TEST2.*
```

shows all the files that were generated during the process (11). EDIT generated the file TEST2. B2S, and the file types .OBJ, .CMD, and .ODL were generated under BASIC control. The LINK command generated the task file, TEST2.TSK, and also the map file TEST2. MAP.

You can copy a file into another directory, with the same group number such as [40, 41] (12). You can also copy files from other directories into your own (13), giving them new names if you wish (14).

After renaming the copies, you can access the files by the name AMORT only if you also specify the other directory (15).

You use a TYPE command to check the contents of the source file and find that the file apparently contains a complete program (16). Then you test it by running the task file (17).

You can list directories other than your own. For example, you can enter the command

```
DIRECTORY [*,*]
```

This requests a listing of all directories in the system (18). You might do this if, for example, you know that someone created a new version of a file on a certain date but are not sure which directory or directories contain it. Once you find the file, there is no need to continue the directory listing, which can be lengthy.

To abort a command that is producing output, you must type CTRL/C (typing C while pressing CTRL) (19). This alerts the terminal for command input. Then you type the ABORT command, specifying the command name DIRECTORY as a parameter, and the directory output stops. A message acknowledges the premature termination (20).

When you type CTRL/C, the terminal immediately stops whatever it is doing and issues the special command prompt, DCL>. This is equivalent to the regular command prompt (>), except that it also reminds you that you have interrupted terminal output. The output continues after you type carriage return unless you enter a complete ABORT command whose parameter is the command or task that initiated the output.

The terminal session is then completed with a LOGOUT command (21).

This illustrative session is only an introduction, intended to acquaint you with the TRAX Support Environment and the command language that you will use to control it. See Chapter 8 for the details of command syntax, and Chapter 9 for a complete description of the DCL command set.

2.3 ACCESSING THE SYSTEM

As you have seen in the sample session, you must begin a session by logging in and identifying yourself to the system. The system manager or operation supervisor at your TRAX installation assigns you a User Identification Code, a User Name, and a password. If you attempt to log in but the system does not recognize your identification, you will be denied access.

2.3.1 User Identification Code

The User Identification Code (UIC) consists of two octal numbers, separated by commas and delimited by a set of brackets. This same code identifies the directory associated with the user. For example, the UIC in the sample session was [40,40], and directories [40,41] and [350,230] were also accessed.

The installation system manager assigns UICs. The first number is a group number, and the second is a member number. Group numbers 1 through 10 are reserved for privileged users. Privileged users have additional capabilities, such as the ability to control other terminals.

The UIC is associated with all of your files and running tasks. It has the same format as a directory specification, and its value is the default directory in file specifications.

2.3.2 The User Name

The User Name consists of an alphanumeric string, 1 to 12 characters long, that identifies you to the system. Each User Name has an associated UIC. The purpose of the User Name is to provide an identification value that is easier to remember than a UIC.

2.3.3 Password

You are assigned a password of your choice as an additional security measure. The password prevents unauthorized access to your files and should be kept secret. The password consists of an alphanumeric string one to six characters long.

2.3.4 Logging In = = the LOGIN Command

After receiving the first Support Environment prompt, you initiate an interactive session by entering a LOGIN command.

```
>LOGIN
```

The system responds by prompting for user identification:

```
USERID?
```

To this you must enter either your User Name or UIC. If you enter your User Name, the system checks its user records to determine the UIC for that User Name. For example, if your User Name is MYNAME, you type:

```
USERID? MYNAME
```

Then the system prompts for the password associated with the UIC and User Name. For example:

```
PASSWORD:
```

You then type your chosen password followed by a carriage return. The purpose of the password is to confirm the user's identity, and it should be kept secret. Thus the password is not echoed at the terminal; that is to say, the characters of the password are not displayed.

If the system recognizes your identification and the password matches the UIC, the system displays an acknowledgement message. This may be followed by a system login message that describes system conditions and anything else that the system manager believes you should know. For example:

```
        TRAX VERSION 1.0A SYSTEM

GOOD AFTERNOON
20-JUL-78 13:59 LOGGED ON TERMINAL TT4:


>
```

However, if either identification value is incorrect, the following error message appears:

```
LOG -- INVALID ACCOUNT
>
```

If you make a mistake during LOGIN, the LOGIN command does not reprompt for either the User Name or the password; you must reinitiate the LOGIN command when the prompt (>) appears again.

After you have completed login, you can proceed with the session.

2.3.5 Terminating a Session = = the LOGOUT Command

To terminate the session, issue the LOGOUT command, as follows:

```
>LOGOUT
```

There are no parameters. The system aborts all uncompleted tasks and dismounts and deallocates any private devices allocated to your terminal. When the system responds with an acknowledging message, the session is over.

```
 TRAX
20-JUL-78 13:57 TT4:  LOGGED OFF
```

2.4 REQUESTING COMMAND INFORMATION = = THE HELP COMMAND

At times you may want to use a command but you do not know its command name, format, or keywords.

The HELP command provides this information.

If you enter a HELP command with no parameters, you will get a complete list of command names.

```
>HELP
   THE FOLLOWING COMMANDS ARE AVAILABLE
  ABORT          ALLOCATE         APPEND           ARCHIVE
  ASSIGN         BASIC            COBOL            COPY
  CREATE         DEALLOCATE       DEASSIGN         DELETE
  DIRECTORY      DISMOUNT         EDIT             INITIALIZE
  LIBRARIAN      LINK             LOGIN            LOGOUT
  MACRO          MERGE            MESSAGE          MOUNT
  PRINT          PURGE            RENAME           RUN
  SET            SHOW             SORT             START
  STOP           TYPE             UNLOCK
  --FOR MORE INFORMATION TYPE 'HELP' FOLLOWED BY THE COMMAND
```

To obtain information about a particular command, include the command name as a parameter. of the HELP command. For example:

```
>HELP SHOW
        SHOW  FUNCTION
                ASSIGNMENTS[:GLOBAL]
                           LOCAL
                DEVICES          **
                DEFAULT
                MEMORY           **
                DAYTIME
                TASKS            **
                TERMINAL         **
                PARTITIONS
                QUEUE            **
```

Notice that some of the keywords are displayed with two asterisks (**). These asterisks are not part of the keyword; rather, they indicate that further information is available regarding the

keyword. To obtain information about a keyword, enter the HELP command with two parameters: the command name and the keyword. For example:

```
>HELP SHOW TERMINAL
        SHOW TERMINAL                   OPTION
                                        TYPE:[NO]SCOPE
                                        LOWERCASE
                                        UPPERCASE
                                        [NO]PRIVILEGED
                                        [NO]REMOTE
                                        [NO]SLAVE
                                        [NO]ESCAPE_SEQUENCE
                                        [NO]HOLD-SCREEN
                                        SPEED:DEVICENAME
```

CHAPTER 3

MANAGING FILES AND VOLUMES

3.1 FUNDAMENTAL CONCEPTS

All information stored in the Support Environment is maintained in logical units called files. A file is a named collection of information organized in a coherent manner. Whenever you wish to store any kind of information–a source program, a body of data, a body of prose text, or whatever, you must have a file in which to store it.

You can create a file at any time with a CREATE command. You can use the flexible EDIT command to create a file and they specify or alter its contents. You can also create files implicitly when using certain of the system facilities; for example, the process of transforming a source language program file into an executable task invariably creates several files along the way.

Once information has been stored in a file, all attempts to access, augment, or manipulate the information must be done in terms of that file. In other words, you must supply a file specification equivalent to the one used to identify the new file.

You, like every other user, have at least one directory, formally called a User File Directory (UFD). A UFD is a special file that serves as an index for all files stored under its auspices. It contains information for each file regarding file identification and the extent to which the file may be accessed.

The magnetic media on which files are stored are called volumes; for example, disks and magnetic tapes. A volume must be mounted (that is, physically positioned, on a device and connected logically to the file system) before you can access any file contained thereon.

3.2 FILE SPECIFICATION CONVENTIONS

A file specification provides all information necessary to identify a file.

```
dev:[ufd] name.type;ver
```

The file specification identifiers are as follows:

dev: Specifies the device on which the volume containing the file is mounted. It consists of either a physical device name or a logical device name assigned to a physical device. Either type of device name contains two alphabetic ASCII characters followed by a one or two digit octal unit number. In a physical device name, the two alphabetic characters constitute a standard device mnemonic known to the system.

A logical device name is defined by an ASSIGN command to be equivalent to a particular physical device name. Table 3-1 lists the standard physical device names for the various devices.

[ufd] Specifies the user file directory (UFD) under which the file is stored. The directory specification is of the form [g,m], where g is the group number and m is the member number. Both g and m are octal numbers in the range 1 to 377. The brackets are required; they identify the information as a directory specification.

name Specifies the name of the file as an alphanumeric string 1 to 9 characters long.

.type Specifies the file type and may serve to identify some aspect of the file's contents. It consists of a period followed by an alphanumeric string 1 to 3 characters long. A period that is not followed by an alphanumeric character constitutes a syntax error. In many cases, files created by system utilities or language processors are given standard file types. See also the section entitled "Standard File Types" later in this chapter.

;ver Specifies the version number of the file, an octal integer to differentiate among files stored in a directory with the same file name and file type. When you create a file, the system assigns the file a version number of 1. If you edit a file, the system keeps the original file for backup and stores the edited file with a version number one higher than that of the original file. If you want to access the latest version of a file but do not know its exact version number, specify 0 or omit the version number entirely. Similarly, you can access the earliest version of a file by specifying a version number of -1.

Table 3-1 Standard Physical Device Names

Device Type	Device Name
Disk	
(RP04/5/6)	DBnn:
(RK07)	DMnn:
(RM02/3)	DRnn:
Line printer	LPnn:
Magnetic tape unit	MMnn:
System default device	SY0:
Logical user terminal	TI0:
Terminal	TTnn:
Virtual terminal	VTnn:

3.2.1 Default File Specification Elements

The file name field is required in every file specification.

Other file specification elements are defaultable. If you do not specify these elements, the system uses the following defaults:

dev: Set up at login time; the default is the device on which the system volume is mounted. That is, the current device associated with the logical device name SYO:. To change this use the SET DEFAULTS command; to display the device, use the SHOW DEFAULTS command.

[ufd] Set up at login time; this default is equivalent to your UIC. To change this use the SET DEFAULT command; to display it, use the SHOW DEFAULT command.

.type Standard file types are used as defaults in some commands, while other commands require explicit file type. The command descriptions in Part II show the default file type, if any, applied to file specifications in each individual command. The leftmost character of the file type is always a period, which is part of the file type. Thus, if a file name is followed by a period no default file type is supplied; if the period is not followed by one to three alphanumeric characters, it constitutes a syntax error.

;ver For input files, the default is the most recent version number; for output files, it is the next higher version number or 1 if no previous version exists.

3.2.2 Wildcards

A wildcard is a special file specification element to allow you to specify a set of files with common elements. By specifying a wildcard (denoted by an asterisk) in a file specification, you can specify more than one file. You may place the asterisk in any file specification field except the device name field. The wildcard causes the system to ignore the contents of the specific field and to select all the files that satisfy the remaining fields.

In general, you can use wildcards in any file specification context that allows multiple files.

For example:

```
>DELETE PROG.CBL;2,PROG.TSK;2,PROG.OBJ;2
```

deletes the three specified files. Since the files have the same file name and version number, but different file types, the following command deletes the same files:

```
>DELETE PROG.*;2
```

Note that if other files exist having the same file name and version number, these will also be deleted. In the case of an output file specification, the system is instructed to replace the field with the corresponding field in the input file specification. As with input file specifications, the device field must not be wild.

3.2.3 Standard File Types

Although you may assign your own arbitrary file types, system operations are simplified by making use of standard file types. The mnemonics listed in Table 3-2 are used by Digital software to reflect actual file contents.

The file type to which the system defaults depends on the command to which the file specification is directed, and on whether it is referring to an input or output file.

Table 3-2 Standard File Types

File Contents Description	Default File Type
Task image file	.TSK
Memory allocation file	.MAP
Symbol definition file	.STB
Object module file	.OBJ
Object module library file	.OLB
Overlay description file	.ODL
Indirect command file	.CMD
Cobol source text file	.CBL
BASIC-PLUS-2 source file	.B2S
Line printer listing file	.LST
Data file	.DAT
System control file	.SYS

3.3 FILE OWNERSHIP AND SECURITY

When you create a file, the system stores it with your UIC in the file header to indicate your ownership of the file. The file is stored with a set of protection codes to indicate who may access the file and for what purpose.

The system also updates a User File Directory (UFD) by adding an entry to reflect the existence of the new file. This entry includes the filename, file type, and version. The directory listing also indicates the file size in blocks and the creation date, and, optionally, the protection code for the file.

3.3.1 The User File Directory

A User File Directory (UFD) is a file that you can create explicitly using a CREATE/DIRECTORY command. The UFD specification is of the form:

```
[g,m]
```

The brackets are required; g is the group number and m is the member number. Both g and m are octal integers in the range 1 to 377. You should consult your system manager to learn the values of g and m that you are allowed to use.

Notice that the UFD specification has the same format as a UIC. Every UIC known to the system normally has a UFD with the same g and m values. This UFD is the default directory; in other words, if you do not include an explicit UFD in a file specification, the system will use the directory associated with your UIC. Thus, if you always default the UFD in your file specifications, all your files will be reflected on the same directory and will be recorded on the system default device. You can list the files stored under a UFD by giving the DIRECTORY command. For example, the following command lists all files stored under the current default directory:

```
>DIRECTORY
```

Besides the information on the directory listing, the directory contains pointers to the header of each file. The file header contains information identifying the owner of the file and the location of the file segments. The following lists all files with the file type CBL:

```
>DIRECTORY *.CBL;*
```

See Part II for detailed description of the CREATE/DIRECTORY and DIRECTORY commands.

3.3.2 File Security

TRAX provides data privacy and system security by a facility that restricts access to volumes and to files contained thereon. The system recognizes four categories of users:

1. **System** users are those with a system UIC. A system UIC has a group number of 1 through 10 octal.
2. **Owner** refers to the owner of the file or volume.
3. **Group** refers to all UIC's with the same group number as in the UIC of the owner.
4. **World** refers to all users of the system, regardless of UIC.

Any of these categories may be truncated to a single letter in the specification; SYSTEM can be written SYS or S, for example.

SYSTEM, OWNER, and GROUP are subsets of WORLD. Any access permission granted at the WORLD level is implicitly given at the GROUP, OWNER, and SYSTEM level. Similarly, OWNER is a subset of GROUP. SYSTEM, however, is a subset of WORLD only, not GROUP or OWNER.

Four types of access are defined: Read, Write, Extend, and Delete. These are specified by the codes R W, E, and D, respectively.

You can specify file protection codes in any command that includes a /PROTECTION qualifier or PROTECTION function. There are four such commands: INITIALIZE, MOUNT, CREATE, and SET PROTECTION. You specify the value of PROTECTION as follows:

```
(category-code: access-code [ , . . . ])
```

For example:

```
>SET PROTECTION A.CBL (SYS:RWE,OWNER:RWED,GROUP:R,WORLD:R)
```

The following rules apply:

1. The parentheses are required.
2. Each user category-code follows a colon. Each category-code may be abbreviated to one or more letters. The colon is immediately followed by the access code.
3. The access-code consists of any or all of the following letters R, W, E, and D, signifying Read, Write, Extend, and Delete access. The letters are given contiguously in this order: RWED.
4. Each category named is given the specific types of access named in its access-code and is denied all types of access not named.
5. Any category not mentioned keeps the access privileges previously assigned to it.
6. Each category-code: access-code string must be followed by a comma or the right parenthesis.

You can specify protection codes for a volume when you initialize it, using the INITIALIZE command. This establishes the primary protection default values for each category mentioned. In the absence of an explicit protection specification, the following default applies:

(SYSTEM:RWED, OWNER:RWED, GROUP:RWED, WORLD:R)

The volume protection specified by the INITIALIZE command can be overridden by a MOUNT command.

Current volume protection codes constitute the default for all files stored on that volume. You can override this default for any individual file, using the CREATE or the SET PROTECTION command.

You can specify file protection when you use the CREATE command to create a file. If you create a file by some other means (such as the EDIT command or as a product of a compilation of LINK procedure), you can modify the protection codes by using the SET PROTECTION command. If you do not specify a protection code for a newly created file, the system applies the file's default code (set at volume initialization).

For example:

```
>SET PROTECTION NEWFIL.CMD (SYSTEM:R,OWNER:RWED,GROUP:RW)
```

modifies the protection rights to the file NEWFIL.CMD as follows:

Access rights of Read to the system category
All access rights to the owner category
Access rights of Read and Write to the group category
Access rights to the world category remain unchanged

Consider the following example:

```
>SET PROTECTION
FILE? NEWFIL.CMD
PROTECTION? (SYSTEM:RW,OWNER:RWD,GROUP:R,WORLD:RWED)
```

In this case WORLD access rights are total. The other three categories are all subsets of WORLD and therefore have all access rights despite the limited rights specified for the individual categories.

You can learn the current protection code on any or all of your files by issuing a DIRECTORY command using the /FULL qualifiers. For example:

```
>DIRECTORY/FULL A.*

DIRECTORY DBO:[40,40]
6-JUL-78 16:49

A.CBL;1                (22653,4)        1./1.              21-JUN-78 13:28
  [40,40]   [RWE,RWED,R,R]
A.ODL;3                (23633,7)        1./5.              21-JUN-78 16:18
  [40,40]   [RWED,RWED,RWED,R]
A.OBJ;3                (23663,5)        2./5.              21-JUN-78 16:18
  [40,40]   [RWED,RWED,RWED,R]
A.TSK;3                (23676,7)        27./27.         C  21-JUN-78 16:19
  [40,40]   [RWED,RWED,RWED,R]     21-JUN-78 16:19(2.)

TOTAL OF 31./38. BLOCKS IN 4. FILES
```

3.4 FILE MANAGEMENT

This section describes the TRAX facilities for creating, manipulating, and listing files.

3.4.1 Creating Files

The Support Environment provides the following methods for creating individual files.

- The RMSDEF utility
- The EDIT command
- The CREATE command

3.4.1.1 The RMSDEF Utility - The RMSDEF utility allows you to define and build command files or define the structure of a data file through an interactive, conversational process. You invoke this utility by entering the following command

```
>RUN $RMSDEF
```

The system responds by asking you a series of questions, prompting for file structure information, and indicating defaults where applicable. If a default is indicated, you can select the default by pressing carriage return. If no default is indicated, you must enter a valid value in response to the question. The following is a typical file definition using RMSDEF.

```
>
>
>RUN $RMSDEF
DO YOU WANT TO GENERATE A COMMAND FILE FOR FUTURE USE(NO)?
ENTER YOUR FILE SPECIFICATION:RDFIL.DAT
FILE ORGANIZATION (SEQ):
RECORD FORMAT (VAR):
MAXIMUM RECORD SIZE (0):512
WILL YOU ALLOW RECORDS TO CROSS BLOCK BOUNDARIES (YES)?
DO YOU WANT CARRIAGE RETURN CONTROL (YES)?
DO YOU WANT PLACEMENT CONTROL (NO)?
ALLOCATION (0 -- IT IS SUGGESTED YOU ENTER A VALUE) :500
DEFAULT EXTENSION QUANTITY (0 - IT IS SUGGESTED YOU ENTER A VALUE):10
DO YOU WANT A CONTIGUOUS FILE (NO)?
SPECIFY PROTECTION BY CLASS
OWNER:(RWED ALLOWED)
GROUP:(RWED ALLOWED)
SYSTEM:(RWED ALLOWED)
WORLD:(R ALLOWED)

YOUR FILE HAS BEEN CREATED!!! -- SY:[40,40]RDFIL.DAT;1

ENTER YOUR FILE SPECIFICATION:^Z
```

After you have provided all the information needed to create a file, the RMSDEF utility informs you that the file is ready to receive input.

NOTE

This utility does not include a facility to let you specify record contents, only the file structure. You write records into the file by means of an application program or the MERGE command.

The utility continues to prompt for file structure information until you terminate it by typing CTRL/Z (^Z). See Appendix A for a detailed description of the RMSDEF utility.

3.4.1.2 The EDIT Command - Issuing the EDIT command invokes the DEC EDITOR. This editor allows you to create a sequential file and specify its contents.

The EDIT command is especially useful for constructing source program files and text files.

The EDIT command is described in Part II of this manual. Also refer to the DEC EDITOR Reference Manual (Order Number AA-5789A-TC) for a complete description of the DEC EDITOR (also known as the EDT Text Editor).

3.4.1.3 The CREATE Command - The CREATE command is a DCL facility provided for users who prefer to specify file structure in a single command rather than in response to a series of prompts. Batch users in particular will usually find the CREATE command easier to use than the RMSDEF facility. The form of the CREATE command is:

```
>CREATE [/qualifier[s] ] file-specification
```

This creates an entry in the appropriate UFD for the file you have specified.

You can create files with any of three types of file organization: relative, sequential, and indexed.

1. SEQUENTIAL files are organized so that records are accessed sequentially. Records are retrieved from a file in the same order in which they were originally written.
2. RELATIVE files are organized so that records of the file may be accessed randomly based on their position relative to the beginning of the file.
3. INDEXED files are organized so that each record has associated with it at least one key field. When records are written to such a file, index tables are constructed. Records are accessed by specifying the part of the record that contains the key.

You specify each by the qualifiers /SEQUENTIAL, /RELATIVE, and /INDEXED respectively. Sequential organization is the default.

The qualifier /FORMAT specifies the record type of the file. Three types are available:

1. *Fixed length records.* All records are equal and non-varying in size.
2. *Variable length records.* At file creation time you specify the size of the largest record that may be written to the file.
3. *Variable length records with a fixed control field.* This record format is supported only for MACRO programmers.

If the created file has indexed or relative organization then you can specify the protection access rights for that file by using the qualifier /PROTECTION. A sequentially organized file takes the default protection set at volume initialization.

For example:

```
>CREATE/RELATIVE/FORMAT:FIXED:120/PROTECTION:(GROUP:RWE,WORLD:RE)
FILE? FILE.DAT
>
```

creates the file FILE.DAT with relative organization and the specified protection code.

If the file organization is sequential, you may type input to the new file line by line following the command string. When a line is terminated, it is sent to the file exactly as formatted at the terminal. You then close the file by typing CTRL/Z.

If the file organization is not sequential, then only a file skeleton is produced. You fill the file either explicitly using a program, or by using the MERGE command.

For indexed files, the command qualifier, /INDEXED/ and KEY are required. When you create an indexed file, you must always specify at least the primary key position and length.

3.4.2 Copying Files

The COPY command creates a sequential file copy of either a concatenated series of sequential files or copies all records from an indexed or relative file.

Copying a disk file to another disk file without specifying file organization (/SEQUENTIAL, /RELATIVE, or /INDEXED) on the input file always produces an exact copy. If, however, the file organization of the input file is specified, the output file is always sequential.

When copying any file to magtape, the output file must always be sequential. Thus, you must always specify the file organization of the input file when copying to magtape. If you copy a contiguous disk file to magtape, and later recopy the magtape copy to disk, the new disk copy will be sequential and noncontiguous.

Multiple file specifications and wildcards are permitted on sequential input files. On indexed or relative input files only one file specification is allowed and wildcards are not permitted.

Since only sequential or identical file copies are created, no file qualifiers on the output file specification are allowed.

Optional qualifiers /REPLACE and /CONTIGUOUS enable you to specify that any previous copy of the output file is deleted before creating a new copy, or that the output file is to be contiguous.

When you copy indexed or relative files, the file structure is not optimized; that is, an exact copy of the file is made.

For example:

```
>COPY ABC.IND/INDEXED/KEY:NUMBER:3
TO? XYZ.SEQ
```

will copy all records from the indexed file ABC.COR to the sequential file XYZ.SEQ in the order specified by the second alternate key.

3.4.3 Appending Records

The APPEND command appends records to an existing sequential file. The records may originate from a collection of sequential files or they may be all the records from within an indexed or relative file.

Qualifiers specify the organization of the input file. If you give no qualifier, sequential organization is assumed.

If you specify (or assume) sequential organization, multiple input file specifications and wildcards are permitted. Since you can append records to only one sequential file, no file qualifiers are necessary (or allowed) on the output file. The same restriction applies to wildcards.

If the originating file is indexed, specify the qualifier /KEY:NUMBER:n and /INDEXED. The default key is the primary key. This will determine the key of access, which in turn determines the order in which the records are accessed.

If the originating file is indexed or relative, only one file specification is permitted and wild-cards are not allowed.

For example:

```
>APPEND PAYROLL.DAT/INDEXED/KEY:NUMBER:2 MASTER.DAT
```

will append all records from the indexed file PAYROLL.DAT to the sequential file MASTER.DAT in the order determined by the first alternate key.

3.4.4 Merging Records

The MERGE command merges records from an input file into an existing output file.

The input file can be either indexed, relative, or sequential (sequential is the default). If the file is indexed, you specify the order of record extraction by using the qualifier /KEY:NUMBER:n. The default key is the primary key. Do not specify wildcards in the input file. The output file must have relative or indexed organization, and must exist before you issue the MERGE command.

The MERGE command can be used to optimize the internal organization of an indexed file. This is useful because an accumulation of updates and deletions to indexed files can cause a fragmented and inefficient index structure in the file.

You must specify the organization of the output file; it is either indexed or relative, but not sequential.

By specifying the optional /LOG, a log of all error messages will be created during the merge sequence. The error messages detail all records that for any reason could not be merged into the output file. They will appear on the terminal or be put into a specified file.

For example:

```
>MERGE PAYROLL1.SEQ PAYROLL2.DAT/INDEXED
```

merges all records from the sequential file PAYROLL1.SEQ to the indexed file PAYROLL2.DAT.

3.4.5 Renaming Files

By issuing the RENAME command, you are able to rename existing files. The specifications of both the original file and the new file must contain both filename and file type. In addition, the device name must be the same in both specifications. Files can be renamed across UFD's, privilege permitting.

For example:

```
>RENAME OLD.TMP NEW.TMP
```

renames the file OLD.TMP to NEW.TMP

3.4.6 Sorting Files

The SORT command invokes the SORT command program, allowing you to read an input file, sort its contents, and write out the sorted data to an output file. Control (or key) fields determine the sorting sequence.

The SORT command also enables you to extract key information, sort that information and store it on a permanent file. This file can then be used to access your original file in the order of the key information on the sorted file.

There are two sorting techniques available:

- *Record Sorting* produces a re-ordered file by sorting entire records on a specified key.
- *Tag Sorting* produces a re-ordered file by sorting only the key records to build a sequence of record pointers.

Specifying the qualifier /PROCESS and the required keyword will invoke one of these sorting techniques.

Alternatively you may specify the qualifier /SPECIFICATION to control and direct the sort. This qualifier has the same effect as /PROCESS but is not limited to sorting files of uniform format.

The other qualifiers to the SORT command define file specifications or other parameters associated with the input and output files.

For full details of the use of the SORT command, read the TRAX SORT Reference Manual. The SORT command is described in Part II of this manual.

3.4.7 Displaying File Contents

The TYPE command displays the contents of all specified files at the terminal. Both the filename and file type are mandatory.

For example:

```
>TYPE MYPROG.CBL,YOURPROG.CBL
```

This displays the contents of two COBOL source program files.

CAUTION

Displaying binary files, such as object task images, can place your terminal in unpredictable modes of operation. If this should occur, see your system manager.

3.4.8 Printing Files

The PRINT command causes one or more specified files to be spooled to a line printer. Spooling is the technique of queuing printer output in the form of jobs; the time a job is actually printed depends on several variables, including a priority value. If you do not specify the file type, .LST is assumed.

The PRINT command has numerous options for controlling the printing of files. See the description of the PRINT command in Part II for further information.

For example:

```
>PRINT MYPROG1,MYPROG2,MYPROG3.TMP
```

prints the files MYPROG1.LST, MYPROG2.LST, and MYPROG3.TMP on the line printer.

3.4.9 Removing Files from a Directory

The DELETE and PURGE commands enable removal of unwanted files from the directory, thereby releasing system resources.

DELETE is oriented to deletion of particular versions of a file, or all versions of a file. In a DELETE command, each file specification must include a file name, a file type and a file version number.

The following command deletes version 3 of the file TEST1.TMP.

```
>DELETE TEST1.TMP;3
```

By specifying version number as a wildcard, you can delete all versions of a file, as in the following example.

```
>DELETE TEST2.TMP;* ITEMS.DAT;2
```

All versions of TEST2.TMP are deleted, and version 2 of the file ITEMS.DAT is also deleted.

The PURGE command also deletes specified files from the directory, but saves one or more of the most recent versions. The following command purges all but the latest version of TEST.TSK.

```
>PURGE TEST.TSK
```

If you want to save more than one recent version, use the /KEEP:n qualifier. For example:

```
>PURGE/KEEP:3 TEST.TSK
```

Assuming, as an example, that the highest numbered version of TEST.TSK is 7, the /KEEP:3 qualifier causes version 7, 6, and 5 to be retained in the directory while deleting all versions of TEST.TSK numbered 4 or less. Notice that you will not necessarily have three versions of TEST.TSK after the purge. If version 6 has been previously deleted, only versions 7 and 5 will remain. Version numbers are always octal integers.

Although the DELETE command syntax requires a version component in each file specification, PURGE file specifications must not include file version numbers.

You can direct DELETE and PURGE commands only to files for which you have delete access rights.

If you specify file names, or versions thereof, that do not exist in the directory, the following error message appears.

```
DEL -- NO SUCH FILE(S)
```

This message is followed by a list of files that you specified for deletion but are not present in the directory. For example:

```
>DELETE TEST.TSK;23,TEST.TSK;4,TEST.TSK;1,TRYOUT.TMP;*
DEL -- NO SUCH FILE(S)
SY0:[40,40]TEST.TSK;23
DEL -- NO SUCH FILE(S)
SY0:[40,40]TEST.TSK;1
DEL -- NO SUCH FILE(S)
SY0:[40,40]TRYOUT.TMP;*
>
```

The error messages indicate that of the files specified in the DELETE command, only TEST.TSK;4 was present and was deleted. The system lists each file specification for which no file or files could be found, along with an error message for each instance.

CHAPTER 4

MANAGING SYSTEM DEVICES AND VOLUMES

This chapter describes the commands and procedures for preparing, assigning, and accessing devices. This chapter describes commands only to the extent needed for you to understand overall procedures. See Part II for detailed command descriptions.

A device is any equipment connected to the system for input and output of information. The most commonly used devices include user terminals, line printers, disk storage units, and magnetic tape units of various types.

At system generation time, all devices in the system are established with certain characteristics, such as line width, speed of data transfer, and accessibility. However, these initial device characteristics are not always suitable for the task at hand, and you may need to make temporary changes to devices in the system. Also, you may need to initialize volumes.

Devices can be either volume-oriented or nonvolume-oriented. Disk or magnetic tape devices are volume-oriented; they store information on interchangeable volumes that can be physically attached or detached from a device. Terminals and line printers are nonvolume-oriented devices; their purpose is to communicate, not store, information.

The volume-oriented devices are file-structured. Volumes must be initialized and mounted before they can be accessed. The nonvolume-oriented devices require no initialization or mounting.

4.1 ACCESSING DEVICES

Access to any given device can be shared by all users, or allocated to one user as a private device. Shared devices are potentially accessible by everyone and are either public or nonpublic. Public devices are not allocatable. Nonpublic devices are available for allocation; when allocated, they become private devices.

As a nonprivileged user (that is, one with an octal group number higher than 10 in your UIC), your file accessing privileges and restrictions are as follows:

1. You can access public devices but not reserve them for your exclusive use.
2. You can access a nonpublic device that is not allocated as a private device by another user.
3. You cannot access a private device allocated to another user.
4. You can allocate a nonpublic device for your private use, if it is available.
5. You can mount volumes only on your private devices.
6. You can learn the status of all devices in the system by entering the SHOW DEVICE command.

4.1.1 Displaying Device Names and Status

The SHOW DEVICES command displays device names, their status, and the system device assignments. The display is made on the entering terminal. The device names appear in the left column while the right column contains information about each device. For example:

```
>SHOW DEVICES
DB0:     PUBLIC   MOUNTED LOADED
DB1:     LOADED
DB2:     LOADED
DB3:     PUBLIC   MOUNTED LOADED
DR0:     LOADED
DR1:     MOUNTED LOADED
MM0:     LOADED
MM1:     LOADED
LP0:     DB0:     SPOOLED LOADED
TT0:     [1,1]    - LOGGED ON      LOADED
TT1:     LOADED
TT2:     LOADED
TT3:     LOADED
TT4:     [40,40] - LOGGED ON      LOADED
TT5:     [1,1]    - LOGGED ON      LOADED
TT6:     LOADED
TT7:     [350,227]        - LOGGED ON      LOADED
TT10:    [1,1]    - LOGGED ON      LOADED
NL0:
VT0:     LOADED
VT1:     LOADED
VT2:     LOADED
TI0:
CO0:     TT0:
CL0:     LP0:
SP0:     DB0:
LB0:     DB0:
SY0:     DB0:
>
```

In the previous list, all device names from DB0 to the TTn names are physical device names. The DBn names indicate disks, MMn indicates magtape drives, LPn indicates line printers, and TTn names indicate physical terminals.

The device names in the left column can be either a physical device name or a logical name. A logical name uses the same syntax as a physical device name. A device name consists of two alphabetic characters and a one or two digit octal number followed by a colon (:).

Beginning with TI0:, the list at the bottom of the SHOW DEVICES example shows the standard pseudodevice names used by some system tasks. Their typical usage is listed.

TI0: for terminal input
CO0: for console output
CL0: for system listing
SP0: for spooling
LB0: for library input device and queue file
SY0: for system input/output device

VT0 is a special device name, indicating a virtual terminal. A virtual terminal is a nonphysical terminal generated by the system at the commencement of each batch job; it provides a terminal environment for processing of the batch job without occupying a physical terminal.

Notice in out example that SP0, LB0, and SY0 are pseudo devices which, in this case, are associated with the same disk unit, DB0.

The following notes describe the device status information that can appear in the right column. More than one message can appear on the same line.

MOUNTED
Indicates that a volume is logically connected to the file system.

PUBLIC
Indicates that the device is a shared device that you can access, but not allocate for private use.

MARKED FOR DISMOUNT
Indicates that the system will dismount the volume when the system completes current file accesses on the volume (no new file accesses may be initiated).

OFFLINE
Indicates that the device was included in the system at system generation time, but for some reason has been removed from the system configuration.

[uic] LOGGED ON
Indicates that the user identified by [uic] is logged onto the system at this terminal.

LOADED
Indicates that the access software for the device is loaded, and the device is available for access.

UNLOADED
Indicates that the access software for the device is loadable, but, is not currently loaded.

SPOOLED
Indicates the device in the left column is a spooled device. When you output files to a spooled device, the system temporarily stores the files on the device specified in the right column. The system transfers the files to the spooled device according to the rules discussed in the Queue Management and Spooling section of the TRAX Manager's and Operations Manual.

Except for spooled devices, a device name in the second column is the physical device for the corresponding logical name in the first column.

A terminal name in the second column followed by the text "- PRIVATE" indicates that the device named in the first column is allocated to the user logged onto the terminal in the second column.

4.1.2 Allocating and Deallocating a Device

Use the ALLOCATE command to establish a specified device as your private device and prohibit other general users access to the device.

You must allocate a device before mounting a volume. This prevents another user from either accessing the volume or allocating its device before you can issue the MOUNT command.

You cannot allocate a public device or a device already allocated. A device already allocated is called a private device.

For efficient resource management, deallocate devices when they are no longer needed. The system manager or the device's owner can deallocate a private device using the DEALLOCATE command. The system automatically deallocates and dismounts your private devices when you LOGOUT.

4.1.3 Mounting a Volume for File Access

The MOUNT command logically connects a volume to a device. The volume must have been previously initialized (see the description of the INITIALIZE command). After you mount the volume, tasks access the volume by specifying the associated device name.

On receipt of a MOUNT command the system verifies that the device is on-line. Also, it checks the volume label that you specify against the volume label on the volume. If the volume labels do not match, the MOUNT command fails.

Each task verifies that a volume is mounted before attempting a file access. This ensures that you access only public devices or your own private devices.

For efficient resource management, you should dismount volumes and deallocate your private devices when you no longer need them. You can mount volumes only on your private devices.

4.4.4 Dismounting a Volume

Use the DISMOUNT command to logically disconnect a volume from the file system. When you issue a DISMOUNT, the system immediately inhibits additional file access by marking the volume for dismount. (As explained in the description of the MOUNT command, the system verifies that file access is permitted before each access.) The system then suspends dismounting the volume if any files are being accessed at the time you issue the dismount command. The system issues a message to your terminal when the dismount operation is complete.

```
12:15:03    *** DB0:   -- DISMOUNT COMPLETE
```

You can dismount only volumes mounted on your private devices. The system dismounts your private volumes and deallocates your private devices when you LOGOUT.

4.2 PREPARING DEVICES

You can alter many device features or characteristics. This section describes the commands required to either initialize, select, display, or change device characteristics.

4.2.1 Displaying and Changing Device Characteristics

The following commands display or change device characteristics:

- SET DEVICES
- SHOW DEVICES
- SET TERMINAL
- SHOW TERMINAL

All devices, such as line printers, terminals, disks, and magnetic tapes, have variable characteristics. These characteristics are given a default value at system startup. You can specify the SHOW DEVICE and SHOW TERMINAL commands to display current device characteristics. The SHOW TERMINAL command displays the terminals' characteristics, while the SHOW DEVICES command displays device characteristics that are applicable to all the devices (including terminals).

You can specify the corresponding SET commands, SET TERMINAL and SET DEVICE, to change characteristics of your private devices.

4.2.2 Initializing a Volume for File Access

The INITIALIZE command produces a file-structured volume on a disk or magnetic tape device.

The system creates a Master File Directory on disk and creates a volume label on magnetic tape. You can re-initialize a volume that was used previously, but the system destroys all existing files on the volume.

After the volume is initialized, you must mount the volume. The volume is then ready for you to access.

4.2.3 Creating a User File Directory (UFD)

The CREATE/DIRECTORY disk creates a User File Directory on the specified device. You are restricted to your private file structured devices.

4.3 ASSIGNING DEVICES

A device name can be either a logical name or a physical device name. The system assigns a device a physical device name during the system generation procedure. A logical name uses the same format as a physical device name and is initially unassigned. Since some tasks use logical device names to access devices, logical names must be assigned to physical devices before such tasks can run. At system start up, the system assigns the system logical names such as SY0: and LB0:. If your installation uses logical names, it is either the responsibility of the programmers to make local

assignments or the responsibility of the system manager to make global assignments of logical names to physical devices before running tasks which depend on such assignments. Using logical names is especially useful if you are not certain which devices are available when you need them. For instance, you can choose a logical name that everyone can use for a certain data pack. Then regardless of where the data pack resides, everyone can access it by using the same logical name.

4.3.1 Making and Changing Device Assignments

Use the ASSIGN command to link a logical name to another logical name or physical device name. Assignments are made at three levels: local, login, and global. When two or three assignment levels specify the same device names, the system resolves the conflicting assignments based upon an established priority. The priority list appears as follows:

1. Local
   Tasks recognize local assignments before login and global assignments. Local assignments apply to tasks executed from the terminal where you made the assignments. You make local assignments with the ASSIGN/LOCAL command. As a privileged user, you can also make local assignments for other terminals by using the /TERMINAL qualifier.

   When you specify the SET DEFAULT command, the system reassigns the login logical name, SY0:, to the local device name you specify. The system accesses the login logical name as the system device, which contains your files.
2. Login
   Tasks recognize login assignments before global assignments. Login assignments apply to tasks executed from the same terminal. The system assigns the login logical name, SY0:, when you issue the LOGIN command. The system assigns the login logical name, SY0:, to the default system logical name, SY0:.

   The system assigns the system logical name to a physical device name at system startup. You can request a display of this information by specifying the SHOW DEVICE command.
3. Global
   Tasks recognize global assignments if there are no local and login assignments. Global assignments apply to all tasks running in the system. You make global assignments with the privileged ASSIGN/GLOBAL command.

Privileged users can assign and deassign any local, login, or global assignment, while nonprivileged users are restricted to the local assignments that they make. Use the DEASSIGN command to remove a local, login, or global logical name assignment.

4.3.2 Displaying Device Assignments

The SHOW ASSIGNMENTS command displays the device assignments. When you specify the command, the system displays your local and login assignments. Assuming that you are logged in at terminal 4, the following example shows an assignment list before and after a local assignment:

```
>SHOW ASSIGNMENTS
SY0:     SY0:     LOGIN   TI - TT4:
>ASSIGN DB2: DO5:
>SHOW ASSIGNMENTS
DO5:     DB2:     LOCAL   TI - TT4:
SY0:     SY0:     LOGIN   TI - TT4:
```

The first SHOW command lists the login assignment. After the ASSIGN command equates logical device name DO5: to physical device name DB2:, this local assignment is shown in the second SHOW command.

4.3.3 Making and Changing Device Assignments

The ASSIGN command equates logical names to physical device names and other logical names assigned previously. You are responsible for local assignments and deassignments. The system makes login and global assignments at system setup time.

The DEASSIGN command disassociates a logical name. There is no automatic deassignment for login assignments when you log off the system. Local assignments can be deassigned explicitly by command or automatically when you log off the system.

CHAPTER 5
PROGRAM DEVELOPMENT

5.1 INTRODUCTION

Depending on the nature of your job and the specific requirements of the installation, you may develop and run your programs in batch or interactive mode. In particular, you might create, edit and test programs interactively and then, after preliminary testing is complete, build procedures and do live runs in batch streams.

You can compile (or assemble), link and execute programs in batch mode using most of the same commands as used interactively; the only difference is that you must add a dollar sign prefix to each command line. For example, a COPY command must be written $COPY if it is to be executed in batch mode. Batch commands are stored in a batch command file before submission to the batch processor using the SUBMIT command. Refer to Chapter 6 for a description of batch processing.

In either interactive or batch mode you may use an indirect command file that contains commands to compile (or assemble), link and run one or more source programs. An indirect command file is a sequential file containing command sequences. To execute the file, in batch or interactive mode, issue an @ sign, followed by the file specification. For example:

```
>@COMFIL
```

The @ sign is used only for invoking indirect command files, and is valid only as the first character in an interactive command line. In a batch command line, the @ sign must be preceded by the $ sign, as with any batch command line.

Generally, you must complete four stages to transform a source program into an executable task and run it. These are:

1. Create one or more source files
2. Translate (compile or assemble) the source file to form an object file
3. Link the object file to form an executable task
4. Run the executable task

The following four sections describe each of the above.

5.2 CREATING SOURCE FILES

In general there are three methods you can adopt to create source files:

1. Invoke the DEC EDITOR facility by issuing the EDIT command. This enables you to create and edit source files. DEC EDITOR is an interactive editing program that uses editor commands to create and modify source programs and other files containing ASCII

character data. Use of the DEC EDITOR is recommended for creating or modifying any type of ASCII file.
2. If you intend to write the source program in BASIC, you can invoke the BASIC-PLUS-2 facility by issuing the BASIC command. This facility provides limited editing functions oriented to the particular needs of BASIC programmers.
3. You can also use the CREATE command, in principle. However, CREATE is designed primarily for creating skeleton files. It includes no editing functions, and is intended primarily for batch processing.
4. You may also set up skeleton files, using the COPY command.

In general, you should use the EDITOR to create BASIC and COBOL source files. DEC EDITOR is an interactive editing program for creating and modifying source programs and other files containing ASCII character data. You can, if you wish, keep skeleton COBOL files containing the division and section headers common to all COBOL programs. Then you can COPY such a file, and use the EDITOR to add the operative information.

5.3 COMPILING SOURCE FILES

TRAX can compile source files written in BASIC or COBOL. The specified source file is compiled or assembled, thus creating an object file. Optional command qualifiers detail the output required. For example, you may request a listing file to be produced.

The /SWITCH qualifier available with COBOL tailors translation of the source file to your particular requirements. The relevant language user's guide contains full details concerning the use of the switches controlled by this qualifier.

5.3.1 Using COBOL

Before running a COBOL program you must create a source file, submit it to the compiler and link the object file. This section details how to compile COBOL programs. Linking and execution is described later in this chapter. For futher details concerning programming in COBOL on TRAX systems consult:

TRAX COBOL Language Reference Manual
TRAX COBOL User's Guide

5.3.1.1 Compiling Source Files – After creating the source file (using either the EDIT command or other suitable facility as described in Section 5.2) issue the COBOL command to compile the specified COBOL source file. More than one source file can be compiled in one execution of the COBOL command.

Command qualifiers detail the output you require. For example, /OBJECT [:object-file-spec] produces an object file named according to object-file-spec. Conversely, /NOOBJECT specifies that an object file will not be produced. /OBJECT is the default.

/LIST [:list-file-spec] produces a listing file named according to list-file-spec. /NOLIST specifies that a listing file is not to be produced. This is the listing facility default qualifier.

The COBOL compiler provides switches that enable you to tailor compilation to your particular requirements. The command qualifier /SWITCHES in conjunction with a particular keyword specify the required switch. The compiler operates according to defaults if you do not specify /SWITCHES. For detailed information on these switches see the TRAX COBOL User's Guide.

For example:

```
>COBOL/OBJECT:PROG1.OBJ  PROG2.CBL
```

compiles file PROG2.CBL to create the object file PROG1.OBJ. No listing file is generated.

5.3.1.2 Linking COBOL Object Files – After you have compiled or assembled the source program and obtained an object file, you must perform one additional step to form the object program into an executable task. This step is called linking.

TRAX is designed to allow routines to access library routines and other user-written routines. All object modules must be processed by the TRAX linker; thus, object files, whether or not they access library or other routines, are not in executable condition as produced by the compiler or assembler.

In the TRAX Support Environment, the task is the fundamental executable unit. A task consists of one or more routines, each routine having been derived from an object file.

As a simple example, assume that you have just compiled a COBOL source program stored in the file COBPROG.CBL. Thus your directory contains an object file with the file specification COBPROG.OBJ. This object file consists of relocatable code; in this conditon, it is called unlinked object module.

The Linker is a system program that takes object modules and system library modules as input, and merges this information to form a task image file. All object modules require references to a system library to determine final storage locations of instructions and data and to establish the required interfaces with the system hardware and software facilities. The Linker resolves these references.

To link the object module COBPROG.OBJ, you could enter the following command:

```
>LINK COBPROG.OBJ,[1,1]COBLIB/LIBRARY,[1,1]RMSLIB/LIBRARY
>
```

This is the simplest instance of the LINK command for a COBOL program using RMS I/O. The name of the task image file defaults to COBPROG.TSK, taking the file name and adding the type .TSK that is standard for task image files.

In this example, the COBOL source program is assumed to contain no CALL statements. In COBOL, the CALL statement is used to reference other user-written routines. If the program did contain a reference to a routine stored in the file EXTMOD.OBJ, that file would also have to be included in the input file sequence:

```
>LINK COBPROG.OBJ,EXTMOD.OBJ,[1,1]COBLIB/LIBRARY,[1,1]RMSLIB/LIBRARY
```

The files COBLIB and RMSLIB are required input file specifications when linking a COBOL program. These are library modules needed to support the COBOL linking. Notice the file qualifier appended to each file specification, indicating to the Linker that they are library modules.

The LINK command has many more optional features to meet various processing demands. In general, command qualifiers further define the action of the Linker and the conditions of the link operation.

Input file qualifiers tell the Linker that some kind of specialized processing is required on the associated input file. For example, /LIBRARY (abbreviated to /LIB in the example above) indicated that the input file contains library modules to be searched before the system library.

The Linker provides an overlay capability as a means of reducing the memory and virtual address space requirements of a task. A task can be divided into overlay segments that reside on disk until they are needed.

The Linker has many optional features and techniques, of which detailed description is beyond the scope of this manual. Part II of this manual describes the LINK command in some detail, defining all the qualifiers. However, you should consult the TRAX Linker Reference Manual for an in-depth description of the Linker.

5.3.2 Using BASIC-PLUS-2

This section provides a general overview of BASIC-PLUS-2 usage; for detailed information regarding the BASIC-PLUS-2 language see the TRAX BASIC-PLUS-2 Language Reference Manual.

The technique of creating executable task images from BASIC source programs is substantially different from that used for COBOL. Before running a BASIC-PLUS-2 program, you must create a file that contains a source program in BASIC-PLUS-2 language. Then you must invoke the BASIC facility to compile the program and prepare it for linking. The BASIC command does not itself compile the program and create the necessary input for the Linker (as COBOL and MACRO does). Rather, it places your terminal under control of the BASIC-PLUS-2 compiler and allows you to direct the compilation. using the command language of BASIC-PLUS-2.

5.3.2.1 Creating BASIC-PLUS-2 Source Files – As mentioned previously, you need not be under BASIC-PLUS-2 control to prepare BASIC source code. However, you may feel it advantageous to have the use of the BASIC-PLUS-2 control language and error detection facilities while preparing BASIC-PLUS-2 source code.

5.3.2.2 Invoking BASIC-PLUS-2 – The following command invokes the BASIC-PLUS-2 compiler:

```
>BASIC
```

There are no qualifiers or parameters.

The system then responds with an identification line followed by the prompt:

```
Basic2
```

This prompt appears whenever the terminal is under BASIC-PLUS-2 control. It follows the invocation of BASIC, and also follows the completion of every BASIC-PLUS-2 control language command. Wherever this prompt occurs, the terminal is expecting BASIC-PLUS-2 input – either BASIC source code or a BASIC control command. It will not accept DCL commands.

5.3.2.3 Compiling and Linking a BASIC-PLUS-2 Source Program – Assume that you have a BASIC source program file MYPROG.B2S in your directory. Before running this program, you must do the following:

1. Issue a BASIC command. After an identification line, the BASIC2 prompt appears, indicating that BASIC-PLUS-2 input is expected.
2. Identify and compile the file you intend to process. For example:

```
OLD MYPROG

Basic2

COMPILE

Basic2
```

   The COMPILE command translates the source file MYPROG into an object module with default filetype .OBJ.
3. Issue a BUILD command. This creates an indirect command file (default filetype .CMD) and an overlay description file (default filetype .ODL). The program is now ready for linking.
4. Issue an EXIT command. The DCL command prompt now appears on the terminal.
5. Issue a LINK command to complete the building of the task. The LINK command (described in more detail in the next section) must include the /BASIC qualifier, and must specify the name of the program specified in the BUILD.

```
>LINK/BASIC MYPROG
>
```

   When the > prompt appears the TRAX linking process is completed. LINK generates an executable task with default filetype .TSK.

   The entire display appears as follows:

```
>BASIC

Basic Plus 2      V01-53
```

```
Basic2

OLD MYPROG

Basic2

COMPILE

Basic2

BUILD

Basic2

EXIT
>LINK/BASIC MYPROG
>
```

When this prompt appears, the linked task file MYPROG.TSK is in your directory and available for use as the parameter of a RUN command.

5.4 TASK EXECUTION AND CONTROL

After you have performed the necessary compilations and linked the object and library modules into an executable task image, you are ready to execute the task, using the RUN command.

You can obtain information about installed tasks by means of the SHOW TASKS command.

Once a task or command has been started, you can stop it and force an orderly termination, using the ABORT command.

5.4.1 Running a Task: the RUN Command

The RUN command directs the system to locate a specified task image file, install it, run it, and remove it from the system following completion. For example:

```
>RUN/TASK:TEST MYFILE.TSK
```

MYFILE.TSK is the file specification of the task image file; it calls for the latest version of the file. /TASK:TEST specifies that the task will have the name TEST while it is in the system.

In this example:

```
>RUN TFILE
```

the task image file is TFILE.TSK; in a RUN command, the filetype default is .TSK. The command gives no explicit task name; thus the system assigns the terminal device name TTnn as the default task name, with nn the unit number of the terminal issuing the RUN command.

5.4.2 Displaying Task Status: SHOW TASKS

The SHOW TASKS command displays information about tasks on your terminal. You can request information on one task only, on all active tasks only, or on all installed tasks. Moreover, you can request a simple list of task names, or detailed information about each task.

5.4.3 Aborting Either a Task or Command: the ABORT Command

The ABORT command terminates the execution of either a RUNning task or the command specified by "task-name". Aborting a task causes the system to force an orderly termination of the specified task. To effect the termination, the system:

- Performs I/O rundown. I/O for all non-file-structured devices are cancelled, I/O for file-structured devices is allowed to complete and then the files are deaccessed. All allocated devices are deallocated.
- Display the abort message.

Upon completion of the ABORT, the task-name is displayed on the originating terminal.

To abort a task started with a RUN command, enter the ABORT command with the command qualifier /TASK. For example:

```
>ABORT/TASK TSK1
```

The task TSK1 is aborted, and a message appears on the terminal when the abort operation is complete.

If the /TASK qualifier is not present in the ABORT command, the system assumes that you are attempting to abort a command. The following two ABORT commands are equivalent:

```
>ABORT/COMMAND TYPE
>ABORT TYPE
```

Prior to aborting any command that produces terminal output (such as TYPE, DIRECTORY, or SHOW) you must type CTRL/C to suspend terminal output. The terminal cannot relay input to the system while producing output.

You can use the ABORT/COMMAND format instead of ABORT/TASK to abort a running task. For example:

```
ABORT RUN
```

This aborts the RUN command that started the running task, and therefore aborts the task itself.

To abort an indirect command file task, enter the following:

```
>ABORT/TASK AT.
```

AT. is the task name applied by the system to all indirect command tasks.

CHAPTER 6
BATCH PROCESSING

The TRAX Support Environment allows execution of commands in either interactive or batch mode, as described previously. In earlier chapters, this manual has described DCL mostly in terms of interactive use. This chapter describes the application of DCL to batch processing.

6.1 FUNDAMENTAL CONCEPTS

A batch file consists of commands and data. Every command line in a batch file must have a dollar sign ($) as its leftmost character and can have a label. Labels allow you to skip commands in the event of a status error. Lines in the file that are not commands are considered data blocks to be used as input to the preceding command.

Every batch file must begin with a $JOB command and end with an $EOJ command. A batch job is structurally similar to an interactive terminal session; the $JOB command is analogous to an interactive LOGIN, the $EOJ corresponds to the LOGOUT, and the data blocks provide the information to the program that you would enter in response to prompts from a running program.

You invoke a batch job by means of a SUBMIT command that specifies an existing batch file in its parameter field. A SUBMIT command can name one or several batch files, and can be given either as an interactive command or as a batch command. That is to say, you can invoke a batch job from another batch job.

Batch jobs are controlled by the Queue Manager. Each batch job is placed in a batch queue, maintained by the Queue Manager to await processing by the batch processor. The batch processor passes individual commands to a command interpreter.

An interactive session is always associated with a physical terminal. Similarly, when you submit a batch job the batch processor creates a virtual terminal. All interaction between the batch jobs and the system facilities is identified with the virtual terminal, thereby freeing the physical terminal (from which the job was submitted) for other use.

6.2 BATCH COMMAND FORMAT

The general format of a batch command is

```
$[label:] command-string
```

Every batch command begins with a dollar sign character. Any batch command line can have a label consisting of 1 to 6 alphanumeric characters followed by a colon. The label allows the command to be the target of a GOTO command that appears earlier in the batch file. Space and tab characters between the colon and the command-string are ignored. The command-string

consists of any DCL command executable by the batch processor. Note that the dollar sign precedes the label.

Batch commands that do not fit onto one line can be continued. A hyphen appears as the rightmost character on the line to be continued. Continuation lines do not start with dollar signs and must not be labeled.

6.3 THE BATCH PROCESS COMMAND SET

Most batch commands are identical to interactive commands, except that the batch command must have a dollar sign prefix and can be labeled. The following interactive commands are not applicable to batch processing:

```
ABORT
ALLOCATE
DEALLOCATE
LOGIN
LOGOUT
```

The following batch commands are not used interactively, or in an indirect command file:

```
$DATA
$EOD
$EOJ
$GOTO
$IF
$JOB
$ON
$SET [NO]ON
```

As mentioned previously, the batch commands $JOB and $EOJ correspond to the LOGIN and LOGOUT commands used to begin and end an interactive session.

The differences between the interactive and batch command sets are due to intrinsic differences between interactive and batch processing.

6.4 THE BATCH LOG FILE

The activities of each batch job are recorded in a log file associated with the job. (NOTE: In this context, "job" means the file or files included in the SUBMIT command that initiated the action.) This provides a hard-copy record of the job similar to the information that appears on a terminal during an interactive session. The log file is printed automatically when the entire batch job is complete unless you specifically ask that it not be printed. You can suppress the log file listing by including the /NOPRINT qualifier in the SUBMIT command that you issue to submit the job. If you specify /NOPRINT, the log file will be placed on your account with a file name corresponding to the job name and a file type of .LOG.

6.5 BEGINNING AND ENDING A BATCH JOB

The $JOB command marks the beginning of a batch job file. It has the following format:

```
$JOB [/TIME:xx] job-name [uic]
```

The optional qualifier/TIME:xx gives the maximum number of minutes (in wall clock time, not CPU time) that the job is allowed to complete its run, beginning from the time the SUBMIT is issued. Job-name is the name by which the job will be identified in the batch log.

The uic parameter specifies a User Identification Code of the form

```
[g,m]
```

The uic is optional. This parameter allows privileged users to log in under another UIC during the batch job.

The $EOJ command marks the end of the batch job file, and has the following format:

```
$EOJ
```

There are no qualifiers and no parameters.

6.6 BATCH DATA BLOCKS

Often you must supply data to programs run under batch control. When you run programs interactively, you enter data in response to prompts from the running task. In batch processing you must supply data in the form of a data block.

In the following example, NEWMEM is an application program that requests information on new members of an organization. It requires the name, address, phone number and dues prepayments for each new member, and uses the information to create new records in a central membership file. You can supply this information to the program by means of a data block. When the end of the data block is reached, NEWMEM terminates, and the running of another application program called MEMLIST supplies a listing of the updated file. The following sequence runs NEWMEM, supplies the necessary data for these new members, and then runs MEMLIST.

```
$RUN NEWMEM
HENRY, SAMUEL
13 OAKLAND AV, SUDBURY
285-9009
20,00
CURRY, JANET
140 LINDENWOOD DR, SUDBURY
287-8123
15.00
```

```
TOWNE, DAVID
NO PERM ABODE
581-3345
0.00
$RUN MEMLIST
```

Notice that the data does not include dollar sign characters at the beginning of any line. Data entered in the form shown above must not include dollar signs as the first nonblank character of the line, because the system would interpret a dollar sign as the beginning of a batch command.

To enter program input that contains dollar sign characters as the first nonblank character on the line, you must precede the input data with the following command:

```
$DATA/DOLLARS
```

This alerts the system that the lines of data to follow may possibly begin with dollar signs. All information that follows this command is treated as data until the following command is encountered:

```
$EOD
```

This command has no qualifiers or parameters; it simply marks the end of the data block. You can include this command at the end of any batch data block, but it is only required when you must terminate data that includes dollar sign characters. ($EOD is also used to terminate data following a $CREATE/DOLLARS command.)

Normally, all data blocks are included in the log file for the batch job. If you wish to suppress this copying, you must include a $DATA command that includes the /NOCOPY qualifier:

```
$DATA/NOCOPY
```

The log file does not receive the data that follows.

In general, you need only include the $DATA command when you wish to use the /DOLLARS qualifer, the /NOCOPY qualifier, or both. $EOD is required only when a prior $CREATE/DOLLARS or $DATA/DOLLARS command is present.

6.7 ERROR STATUS AND SEQUENCE CONTROL

Commands and tasks return a status on exit, indicating whether an error occurred. In batch processing you can specify alternative action to be taken in the event of an error.

6.7.1 Status Levels

Any of four exit status levels can occur:

SUCCESS
WARNING
ERROR
SEVERE_ERROR

If exit with status is not implemented in the task or command, no status level is returned to the batch processor and execution continues as if the status had been SUCCESS.

SUCCESS indicates that results should be as expected.

WARNING indicates that the task has succeeded, but with possible irregularities, and that results may not be as expected.

ERROR is stronger than WARNING; results are unlikely to be as expected.

SEVERE_ERROR indicates one or more fatal errors and that the command or task may have been terminated prior to completion.

6.7.2 Conditional Processing

Four batch commands are designed to control the processing sequence in a batch procedure. They specify alternative action to be taken by the batch processor should an error occur in a command or task.

6.7.3 The $ON Command

The $ON command specifies action to be taken in the event that any subsequent command returns an exit status with a severity as great or greater than that specified in the command. Its format is:

$ON status-level THEN action

The status-level must be one of the following:

WARNING
ERROR
SEVERE_ERROR

Then the action must be one of the following:

CONTINUE
STOP
GOTO label

The arguments of the $ON command are stored in local memory, and referenced whenever a command or task that returns a status level is executed. $ON is a global command. These arguments remain in force until superseded by another $ON command, until end-of-job, or until the $ON command actually takes effect. Any individual $ON command can be executed only once.

If no $ON command is in effect, and execution produces an exit status of ERROR or SEVERE-ERROR, the processing of the batch job stops. That is to say, the initial (or default) setting is $ON ERROR THEN STOP. The STOP action causes the batch processor to skip all remaining commands in the batch file. If an $ON command is found, on attempted execution, to be faulty, the batch processor reverts to the default setting.

Example:

```
$ON ERROR THEN STOP
$COBOL MYPROG
$LINK MYPROG
$RUN MYPROG
```

If the assembly is completed with a status of success or warning, the job continues with the linking. If the linking produces no status worse than a warning, the task is run. If however, a status level of ERROR or SEVERE-ERROR is produced by the $MACRO, $LINK, or $RUN command, the batch job is stopped.

6.7.4 The $SET [NO]ON Command

The $SET NOON command suspends the influence of the $ON command currently in effect. Its format is:

```
$SET NOON
```

The $SET ON command reinstates the $ON command that was previously negated by a $SET NOON command. Its format is:

```
$SET ON
```

6.7.5 The $IF Command

The $IF command is similar to the $ON command, except that it operates locally, pertaining only to the last preceding command (excluding other sequence-control commands). Also, it tests only for status-level actually specified; the THEN action is executed only if the status-level returned by the preceding command matches exactly the status-level specified in the $IF command. Its format is:

```
$IF status-level THEN action
```

Status-level must be one of the following:

SUCCESS
WARNING
ERROR
SEVERE-ERROR

The action parameter is the same as for the $ON command.

6.7.6 The $GOTO Command

The $GOTO command instructs the batch processor to unconditionally skip all commands up to a specified label. Execution continues at the command bearing that label. Only forward branching is allowed. The format is:

$GOTO label

The $GOTO command must appear with a label. The label must appear, followed by a colon, in front of a later command, or the job is terminated. For example:

```
$ON WARNING THEN GOTO ELSE
$LINK MYPROG
$RUN MYPROG
$GOTO END
$ELSE: ON WARNING THEN GOTO END
$LINK OLDPROG
$RUN OLDPROG
$END: EOJ
```

In this example, MYPROG is linked and run unless the link includes warning errors or worse, in which case OLDPROG is linked and run. If linking OLDPROG results in an error status, the setting of ON causes the command processor to look ahead for the label END.

Note that GOTO can be used both as a standalone command and as part of an $IF or $ON command. Both types of usage are depicted in the preceding example.

If no warning status occurs during the linking of MYPROG, MYPROG is run. On completion, the batch processor skips ahead to the terminating EOJ command.

6.8 SUBMITTING A BATCH JOB

You can submit a batch job during an interactive session or from another batch job with the following command:

[$] SUBMIT [/qualifiers] batch-file-spec [,...]

The batch-file-spec is promptable. Each batch-file-spec must refer to a file that consists of batch commands, the first command being a $JOB command. If no file type is given, the default is .CMD.

The SUBMIT command places the batch file or files into a batch queue to await processing.

Qualifiers allow you to specify:

- The batch queue into which the job is to be placed (/QUEUE)
- Whether the job can be restarted from the beginning following an interrupt (/[NO] RESTART)
- The queue priority for the job (/PRIORITY)
- Printing or non-printing of the log file for the job (/[NO] PRINT)
- A date and time at which the job will become eligible to be dispatched from the batch queue to the batch processor (/AFTER)
- The name of the batch job (/JOB)
- Whether to submit original copies of files from a private volume, or to make a temporary copy ([NO] ORIGINAL)

CHAPTER 7

INDIRECT COMMAND FILES

In using the Support Environment, you may find that you use certain interactive command sequences (or long single commands) fairly often. Rather than type such commands each time you want to execute them, you can store them in an indirect command file.

An indirect command file is a sequential file that contains one or more interactive commands. The commands in an indirect command file are invoked as a unit, and are executed as single commands, one after another, until you reach the end of the file. Unlike batch files, they execute immediately at your terminal, and can accept interactive input.

7.1 CREATING AN INDIRECT COMMAND FILE

You can create an indirect command file most easily by calling the DEC EDITOR, using the EDIT command. (See the TRAX Text Editor Reference Manual for information on the use of the DEC EDITOR.) The standard .CMD file type is recommended. Then enter commands as text in the same format as you would enter them on interactive session. For example:

```
COPY [350,230]TRANSACT.DAT TRANSBACK.DAT
PURGE TRANSBACK.DAT
PRINT TRANSBACK.DAT
```

You can, if you wish, create an indirect file for generalized use, omitting the command parameters. When you invoke the file, the individual commands will prompt for its parameters. For example:

```
COPY
PURGE
PRINT
```

This indirect command sequence will prompt for parameters at your terminal on each command, so that you can use it for any file. This technique is especially useful for creating sequences that include lengthy commands with fixed qualifiers, but variable parameters.

7.2 INVOKING INDIRECT COMMAND FILES

To execute indirect command files, enter a command consisting of an at sign, @, followed by file specification of the file containing the commands.

If the file type is omitted, the default file type .CMD is assumed. Thus the following two commands are equivalent:

```
@INDSEQ

@INDSEQ.CMD
```

You can invoke an indirect command file in either interactive mode or batch mode, but the indirect command file must contain only interactive commands. To invoke the file INDSEQ from a batch file, the command is:

```
$@INDSEQ
```

Illustrative Applications

1. Suppose you have an indirect command file IND.CMD that contains the following information:

   ```
   COPY
   PURGE
   TYPE
   ```

   You invoke this file by entering the following interactive command.

   ```
   >@IND.CMD
   ```

   The three commands prompt for parameters and execute as follows:

```
>@IND.CMD
>COPY
FROM? [350,230]AMORT.B2S
TO? REVAMPT.B2S
>PURGE
FILE? REVAMPT.*
>TYPE
FILE? REVAMPT.B2S
10        input 'interest' j
20        let j=j/100
30        input 'amount' a
40        input 'number of years';n
50        input 'payments per year'; m
60        let n=n*m \ i=j/m \ b=1+i
70        let r=a*i/(1-1/b^n)
100       print'amount per payment=';int(r*10^2+.5)/10^2
110       print 'total interest      =';int((r*n-a)*10^2+.5)/10^2
1000      end
>@ <EOF>
>
```

Note that you enter only the @ command and the responses to the prompts for file specification. After the last command is executed, the system indicates this by displaying the following:

```
@ <EOF>
```

2. The following sequence deletes all copies of temporary files (with file type .TMP) and purges all but the most recent copy of all other files prior to logout.

```
DELETE *.TMP;*
PURGE *.*
LOGOUT
```

This kind of directory cleanup is often a lengthy procedure. By using an indirect command file, you can invoke the procedure and allow it to run without attending the terminal. The terminal listing is as follows:

```
>@LOGF.CMD
>DELETE *.TMP;*
>PURGE *.*
>LOGOUT
15:23:28    TASK "AT.T4 " TERMINATED
HAVE A GOOD AFTERNOON
        ABORTED VIA DIRECTIVE OR MCR
03-JUN-78 15:23 TT4:  LOGGED OFF
>
```

You need not leave the terminal power up once you have invoked the indirect command unless you need the listing.

PART TWO

SUPPORT ENVIRONMENT COMMANDS

– REFERENCE –

CHAPTER 8
FORMAT CONVENTIONS

8.1 COMMAND DESCRIPTIONS

Command descriptions in Chapter 9 include the following information, as required:

1. The name of the command.
2. A brief statement of the command's purpose or action.
3. The general format of the command, showing all elements of the command. See Section 8.2 for detailed information on command format.
4. The possible prompts that the command can issue to request additional information, such as file specifications or tasknames omitted from the command.
5. Descriptions of the command parameters, such as file specifications.
6. Descriptions of the qualifiers for the command. Qualifiers are key words whose first character is a slash (/). Generally, there are two categories of qualifiers: command qualifiers and parameter qualifiers.

   Command qualifiers are those which appear immediately after the command name, before any parameters that the command may contain. They influence the overall command action. Many commands allow or require multiple qualifiers.

   Parameter qualifiers influence only the parameter whose specification they immediately follow.
7. Whatever additional notes may be needed to describe the syntax and action of the command.

8.2 GENERAL FORMAT NOTATIONS

In describing general command formats, notation conventions are as follows:

1. Command names are shown in upper-case letters. Batch command names also have a leading dollar sign ($).
2. Parameters are shown in lower-case letters, and specify the type of information that you must provide. Parameters used in the descriptions are as follows:

Parameter	Information Required
file-spec	File specification, as described in Chapter 3.
in-file-spec out-file-spec	The parameters in-file-spec and out-file-spec are used only when needed to indicate the order of the parameters (as in a COPY command).
task-name	The name of an executing task.
function	A secondary keyword required in some commands to further define the action of the command.

device-name	A device designation; this can be either physical device name or a logical device name.
logical-name	A device designation; this must be a logical device name.
ufd	A User File Directory.
user-id	A User Identification Code or User Name.

The above list is not complete. Descriptions of some specialized parameters are included in the description of the command in which they appear or are self-explanatory.

3. Qualifiers are depicted in two forms:

   - A slash followed by a string of upper-case letters, indicating the actual characteristics of the qualifiers.
   - A slash followed by the lower-case word "qualifier", indicating that any of several qualifiers are valid.

4. Colons, periods, commas, semicolons, dollar signs, and slashes are part of the elements in which they appear, and are required where shown.
5. Square brackets [ ] indicate that the material enclosed within is optional.
6. Ellipses ... indicate that the immediately preceding parameter is repeatable; i.e., multiple values are allowed for the parameter. Separate multiple values by commas.

8.3 ISSUING COMMANDS

You communicate with the system by issuing commands. A command consists of a command name which describes the action the system is to take (COPY or LOGIN, for example), often accompanied by one or more parameters. Parameters either describe the items on which the command is to act or further define the function of the command.

Both command names and parameters can have qualifiers. Qualifiers are appended as a suffix to modify or further define the command name or parameter.

Commands can be entered at an interactive terminal only when the system is prompting >. Some commands (EDIT and BASIC, for example) invoke a program that accepts its own set of commands, valid only while that program is running. In turn, system commands are not valid while that program is running; you must first return control to the system. The descriptions of EDIT and BASIC in Part II describe how to terminate the invoked program's execution.

8.3.1 Command Structure

A command consists of a command name followed by a set of parameters. (A batch command also contains a dollar sign and an optional label.) The command name specifies the type of action for the system to perform, and the parameters indicate those entities on which the command will perform the action.

A command name or parameter can include a set of qualifiers that modify or complete its meaning. Qualifiers are appended directly to the command name or parameter with no embedded spaces.

The general format of a command is as follows:

```
[$[label:]] command-name [/qualifier...] parameter [/qualifier...]...
```

You can either supply the command name followed by the parameters on one line or enter the parameters in response to prompts. In both batch and interactive mode, when two or more parameters are on one line, they must be separated by at least one space or tab.

A command may require more than one line A hyphen (-) as the last character on the line continues the command onto the next line. Following the carriage return, the system prompts:

```
DCL>
```

An exclamation mark (!) line indicates the start of a comment. The comment text appears after the exclamation mark, and the rest of the line is treated as commentary.

8.3.2 Command Names

Every command begins with a command name that describes the action the command is to perform. You need not enter the complete command name to have it function correctly, only enough of its leading characters to uniquely identify it. No command name requires more than four characters for unique identification, and in some cases one character suffices.

For example, DEASSIGN and DEALLOCATE can be abbreviated to DEAS and DEAL respectively, but further abbreviation would make the command ambiguous. However, TYPE can be abbreviated to T, because no other command begins with that letter.

NOTE

To ensure compatibility with further versions of TRAX (which may include new commands), you should use at least four characters to identify commands in batch and indirect command files.

8.3.3 Parameters

A parameter either describes a value that a command uses when executing, or it further defines the action of the command. At least one space or tab must separate the first parameter from the command name; parameters are then separated from each other by one or more spaces (and/or tabs).

If you do not enter all the parameters that the system requires to execute the command, the system prompts until all required parameters are entered.

As an example, the COPY command is used to make new copies of files. It requires the name of the file to be copied and the name of the new file to be created. If only the command name, COPY, is entered, the system prompts for the name of the existing file by typing FROM?. It then prompts for the name of the file to be created by prompting TO?.

The COPY command can be entered in any of the following ways.

```
>COPY OLDFILE.DAT NEWFILE.DAT

>COPY OLDFILE.DAT
TO? NEWFILE.DAT

>COPY
FROM? OLDFILE.DAT
TO? NEWFILE.DAT
```

8.3.3.1 **Optional Parameters** - TRAX DCL never prompts for optional parameters. You must supply optional parameters on the same line with required parameters. This means that you must respond to each parameter prompt appearing on your terminal. For example:

```
>SHOW
FUNCTION? QUEUE ALL
  PRINT QUEUES
    PRINT
    LPQO
    TEST
    BAPO
  BATCH QUEUES
    BATCH
    TRXKIT
    SURVEY
    CHRIS
```

8.3.3.2 **Parameter Lists** - Some commands allow a parameter to be replaced by a list of parameters. For example, in a DELETE command, a single file specification can be replaced by a list of file specifications. When a parameter is a list of items, the items are separated by commas. Extra spaces are ignored.

Example:

```
>DELETE ABC.CBL;*,    AB.OBJ;1
```

8.3.4 Qualifiers

A qualifier consists of a character string, recognizable by the system, with a slash (/) as its first character. Its purpose is to modify or further specify the meaning of a command name or a parameter.

Qualifiers are directly appended to the qualified element, so that an element and all of its qualifiers form a string with no embedded spaces. Any qualifier may be abbreviated if it contains enough characters to distinguish it from any other possible qualifiers for that command. Any qualifier can be uniquely abbreviated to four leading characters.

8.3.4.1 **Command Qualifiers** - Command qualifiers modify the action of a command. For example, consider the following COBOL command string:

```
>COBOL/NOOBJECT COBSRC.CBL;1
```

Normally (i.e., that is, by default) the COBOL command produces an object file. In the example string, the /NOOBJECT qualifier overrides the default action, and no object file is produced.

Some commands have no qualifiers, while others have many possible qualifiers. Multiple qualifiers are permitted. For example:

```
>COBOL/NOOBJECT/LIST:COBLST.TMP  COBSRC.CBL;1
```

Some command qualifiers include variables; the /LIST qualifer, for example, can specify a file to receive a listing. There is no prompting for qualifiers or for qualifier variables.

8.3.4.2 **Parameter Qualifiers** - In some commands, parameters such as file specifications can have qualifiers. Parameter qualifiers further specify parameters that require special treatment. In the following APPEND command, the /RELATIVE qualifier informs the system that the input file DAT.TMP;1 has relative file organization.

```
>APPEND DATA.TMP;1/RELATIVE  UPDATE.DAT
```

8.3.5 Underline Convention

To improve readability, some DCL words include an underline character joining two or more English words. For example:

```
CREATE/VOLUME_LABEL    SEVERE_ERROR
```

When such DCL words are abbreviated the underline is treated the same as an alphabetic character. Thus the following are all valid abbreviations for the qualifier VOLUME_LABEL:

```
VOLUME_L
VOLUME_
VOLU
```

The following abbreviations are not valid:

```
VOLUME_LBL (interior characters omitted)
VOLUME-LAB  (hyphen vot valid)
```

8.4 TERMINAL KEYBOARD FUNCTIONS

You type the input text one line at a time, terminating each line with a carriage return (RETURN). The system either prints the terminal input on the terminal printer or displays it on the screen of a display unit.

Function keys can be used to format a line (Space Bar, TAB), to edit a line (DELETE), or to access the uppermost of two characters that appear on a key (SHIFT, SHIFT LOCK). Typing a carriage return (RETURN) causes the system to process the current line.

Table 8-1 describes the function keys, as they appear on the LA36 and VT52 support terminals, and the effects of their use.

The CTRL key produces a variety of functions when pressed simultaneously with any one of several letter keys.

The combination of CTRL and another character key is called a control character. In this manual a control character is written as "CTRL/x" where x represents a variable character key.

The effect of a control character sometimes depends on the activity that the terminal is currently supporting.

Table 8-2 lists all the control characters supported under TRAX and their associated functions.

Table 8-1 Keyboard Functions

Key	Description
CTRL	Used in combination with several 1-key letter keys to produce a variety of functions.
DELETE	Deletes the last character typed at the terminal, and further continuous characters if the key is pressed repeatedly.
ESC	Terminates a line of input without moving the carriage or cursor.
LINE FEED	Physically moves the paper or rolls the screen image upward, without influencing the system in any way.
RETURN	Terminates the current input line and enters the line into the system; the carriage or cursor advances to the first character position of the next line.
SHIFT	Prints or displays the uppermost of two characters appearing on a key typed while SHFFT is held down.
SHIFT LOCK or CAPS LOCK	Alternately locks and unlocks SHIFT mode on alphabetic characters. This key does not affect nonalphabetic characters.
SPACE BAR	Advances carriage or cursor one space at a time.
TAB	Causes the carriage or cursor to move to the next tab stop on the line. A line conventionally contains tab stops every eight character positions.

Table 8-2 Control Key Functions

Key	Description
CTRL/C	CTRL/C typed either as the first character in the line or when the terminal is producing output causes the system to prompt for command input. If the last character entered at the terminal was CTRL/S, CTRL/C also performs the function of CTRL/Q.
CTRL/I or TAB	Advances the carriage or cursor to the next horizontal tab stop on the line. The system establishes tab stops at every eight characters in the line.
CTRL/K	Causes a vertical tab by performing four line feeds.
CTRL/L	Performs eight line feeds.
CTRL/O	Alternately suppresses and resumes the display of output at the terminal. The system discards characters directed to a terminal that has disabled the display of its output.
CTRL/Q	CTRL/Q typed after a CTRL/S resumes output suspended by the previous CTRL/S.
CTRL/R	Typing CTRL/R before typing a line terminator causes the system to retype the current line on a new line, omitting any depleted characters. If the current line is empty, CTRL/R performs a carriage return and line feed
CTRL/S	Typing CTRL/S while the terminal is receiving output suspends additional output until you type CTRL/Q or CTRL/C. The suspended output is merely delayed, not lost (see the description of CTRL/Q). The combined functions of CTRL/Q and CTRL/S are convenient when using a display terminal that transmits faster than you desire.
CTRL/U	Typing CTRL/U before typing a line terminator causes the previously typed characters to be deleted back to the beginning of the line. The system responds with a carriage return and line-feed so that the line can be re-typed. CTRL/U is echoed as CU.
CTRL/Z	Is a break character indicating end-of-file. Use it when the system is expecting input as a signal to indicate that you are finished typing in data. Most system utilities will bring all processing to an orderly termination and exit in response to this function.

8.5 CORRECTING INPUT ERRORS

Before terminating a line, you can correct typing errors by using the DELETE key or change the line completely by using CTRL/U. However, once a command has been terminated (and thus input to the system) it cannot be corrected; the system will perform, or attempt to perform, the operation you specified. File information can be corrected by editing.

8.5.1 Deleting Individual Characters

The DELETE key deletes the most recent character on the current line for each pressing of the key. DELETE has no effect when the current line is empty.

On a hard-copy terminal, each deleted character is echoed. The string of deleted characters is enclosed between an initial and a final backslash (\). These backslashes are generated by the system for visual reference, and are not included in the data. The final backslash is printed when a new text character is typed in place of DELETE. Backslashes and deleted characters are omitted in the case when CTRL/R is used to make a copy of the line as typed so far.

On a video terminal, such as a VT52, each deleted character is removed from the screen, and the cursor returns to where it was before the character was typed.

For example, to change ACCDE to ABCDE, the user presses DELETE four times to override the CCDE. On a hard-copy terminal the string now appears as

```
ACCDE\EDCC
```

The user then enters the correct sequence BCDE. On the hard-copy terminal, the string now appears as

```
ACCDE\EDCC\BCDE
```

On a display unit the screen will show the string

```
ABCDE
```

In both cases ABCDE is the string accepted and sent to the computer when the line is terminated.

8.5.2 Deleting a Line

CTRL/U deletes all characters on the line, prints ^U, and performs a carriage return. The user can then enter the text correctly.

For example, if you type ACCDEFGHI, but meant to type B for the first C, pressing the DELETE key eight times would be tedious and the result confusing on a hard-copy terminal. It would be easier to press CTRL/U and start again. The latter method would appear as follows:

```
ACCDEFGHI ^U
ABCDEFGHI
```

After using the DELETE key to correct a line and before terminating the line, you can ensure that the final result is what you want by typing CTRL/R before pressing carriage return. This displays the connected line contents before it becomes computer input.

Further corrections can be made at this point if necessary.

8.6 ABBREVIATIONS

When you type a system keyword (such as a command, or qualifier, or fixed parameter value), you need only type enough leading letters to uniquely identify it. However, the characters that you do include must match those of the corresponding characters in the full name. The system does not attempt to resolve invalid names by dropping characters from the right-hand end. If, for example, you are typing a COBOL command:

COBOL	is a complete command name.
COB	is a valid abbreviation of COBOL.
CO	is ambiguous, because COPY begins with the same letters.
CBL	is invalid, because no command begins with the letters CBL.

Any keyword can be abbreviated to the first four letters and be recognizable. Some keywords can be abbreviated to a single letter. Nevertheless, when creating batch procedures or indirect command files for long-term use, you should consider limiting abbreviations to four characters; this will ensure compatibility with future enhancements.

CHAPTER 9
COMMAND DESCRIPTIONS

This chapter describes the set of TRAX commands available to the general user. Commands are presented in alphabetical order.

9.1 ABORT

The ABORT command terminates the execution of a command or a running task that you have originated. Aborting a task causes the system to force an orderly termination of the specified task. To effect the termination, the system:

- Performs I/O rundown. I/O for all nonfile-structured devices is cancelled. I/O for file-structured devices is allowed to complete and then the files are deaccessed. All attached devices are detached.
- Displays the abort message.

Upon completion of the ABORT, the task name is displayed on the originating terminal.

Format:

ABORT[/qualifier] task-name

Command Qualifiers:	*Default:*
/COMMAND	/COMMAND
/TASK	
/DUMP	

Prompts:

```
>ABORT
COMMAND NAME?
```

or

```
>ABORT/TASK
TASK NAME?
```

Command Parameter:

task name	The name of either the task or command to be aborted. If a command name, it may be abbreviated. See the Command Qualifiers for further information.

Command Qualifiers:

/COMMAND	Abort a DCL command; such as DIRECTORY or RUN. The task-name given to an interactive command is XXXTnn where XXX is the first three characters of the command and nn is the number of the originating terminal. If the command is a batch command (originating from a virtual terminal), a V appears in the task-name instead of a T.
/TASK	Abort a user task. ABORT/TASK "name" is used to abort a task whose name appears in the active task list for the issuing terminal or for the system as a whole. The indirect command file task is aborted by specifying ABORT/TASK AT. on the originating terminal. See the SHOW TASKS command to display active task names.
/DUMP	Requests a post-mortem dump of the aborted task or command.

Notes:

If the command or task that you want to abort is producing output (as in the case of a DIRECTORY command, you must type CTRL/C. This causes the system to prompt DCL>. Then you enter the complete ABORT command, including the command or task-name. If you type ABORT and then press carriage return, the system will not prompt until it has completed its current output commitment. If you are issuing the ABORT command to suppress unwanted terminal output, you must enter the complete command in response to the DCL> prompt.

Examples:

This example aborts the currently active user task called TEST.

```
>RUN/TASK:TEST ENDLES.TSK

DCL>ABORT/TASK TEST
11:38:35    TASK "TEST   " TERMINATED
            ABORTED VIA DIRECTIVE OR MCR
```

This example aborts the indirect file processor task from the issuing terminal (TT4:).

```
>@LOGF

DCL>ABORT/TASK AT.
>DELETE *
11:42:40    TASK "AT.T4 " TERMINATED
            ABORTED VIA DIRECTIVE OR MCR
            AND WITH PENDING IO REQUESTS
```

9.2 ALLOCATE

The ALLOCATE command establishes a specified device as a private device and denies other general users access to the device.

You must allocate a device before mounting it. For efficient resource management, devices should be deallocated when they are no longer needed. (Refer to the DEALLOCATE command for its use.) Only the system manager or the device owner can deallocate a device. The system automatically DEALLOCATEs your private devices when you log off (LOGOUT) the system.

Public devices or other users' private devices cannot be allocated.

Format:

ALLOCATE device-name

Prompt:

DEVICE? device-name

Command Parameter:

device-name	The device name of the device to be allocated. A list of device types is provided in Chapter 3, Table 3-1.

Command Qualifier: None.

Examples:

This example allocates the DB2: disk. Other users are not permitted to use DB2: until you or the system manager deallocates it.

```
>ALLOCATE DB2:
```

9.3 APPEND

The APPEND command copies one or multiple sequential files, or the records of one relative or indexed file, to the end of an existing sequential file.

Format:

APPEND input-file-spec[/file-qualifier] output-file-spec

File Qualifiers:	*Default:*
/SEQUENTIAL	/SEQUENTIAL
/RELATIVE	
/INDEXED	
/KEY:NUMBER:n	

Prompts:

FROM? input-file-spec[/file=qualifier] . . .
TO? output-file-spec

Command Parameters:

input-file-spec, . . .	The file specification of the input file or files. If multiple file specifications are given , the entire set of specifications must be enclosed in parentheses.
output-file-spec	The file specification of a sequential file to which the records of the input files will be appended. All file specifications must include a file name and a file type.

File Qualifiers:

/SEQUENTIAL	Specifies that the input file or files has sequential organization. This is the default.

/RELATIVE	Specifies that the input file has relative organization.
/INDEXED	Specifies that the input file is organized as an indexed file.
/KEY:NUMBER:n	Optionally specifies the record access key for an indexed file. If n=1, it calls for access on the primary key defined for the input file. If n=2, it specifies access on the first alternate key; n=3 specifies the second alternate key, and so on, up to the number of keys defined for the input file.

Notes:

1. The output file must have sequential organization.

2. Wildcards are allowed on input files with sequential organization only. The order of appending multiple files specified by wildcard is their order of appearance in the directory and can be seen in advance by issuing a DIRECTORY command using the same wildcard specification.

3. If the input file is organized indexed or relative, the APPEND command must include the appropriate qualifier.

Example:

This example appends the contents of FILE 1. DAT to the end of FILE 2. DAT.

```
>TYPE FILE1.DAT
THIS IS FILE 1.
>TYPE FILE2.DAT
THIS IS FILE 2.
>APPEND FILE1.DAT
TO? FILE2.DAT
>TYPE FILE2.DAT
THIS IS FILE 2.
THIS IS FILE 1.
>
```

9.4 ASSIGN

The ASSIGN command defines logical device names.

Format:

ASSIGN[/LOCAL] device-name logical-device-name.

Prompts:

DEVICE? device-name
LOGICAL? logical-device-name

Command Parameters:

device-name	The physical device name of the device or a previously assigned logical name.
logical-device-name	The logical name to be assigned.

Command Qualifier:

/LOCAL	Optional, default

Notes:

1. The logical-device-name consists of a two-character ASCII string followed by a 1- or 2- digit octal number and a mandatory colon.

2. The assignment continues in effect until you either use a DEASSIGN command specifying the logical name or logout. No automatic deassignment occurs when you dismount the the physical device.

Example:

When the COPY command is executed, the value of the logical device name, TA5: is replaced by the actual physical device name MM1:.

```
>ASSIGN MM1: TA5:
>COPY TA5:*.* *.*
```

9.5 BASIC

The BASIC command invokes the BASIC-PLUS-2 compiler and places the terminal in BASIC-PLUS-2 control mode.

Format:

BASIC

Notes:

1. The BASIC command has no qualifiers or parameters. After you enter the BASIC command, the following information appears:

   BASIC 2

   This indicates that you are in BASIC-PLUS-2 mode and must enter only commands appropriate to that mode when handling files.

2. To exit BASIC-PLUS-2 and return to TRAX, enter the following command in response to a BASIC 2 prompt.

   EXIT

3. See BASIC-PLUS-2 documentation for information about the BASIC-PLUS-2 programming language and control commands.

9.6 COBOL

The COBOL command compiles one or more COBOL source program files.

Format:

COBOL[/qualifiers] file-spec [,...]

Command Qualifiers:	*Default:*
/LIST[:file-spec]	/NOLIST
/NOLIST	
/OBJECT[:file-spec]	/OBJECT
/NOOBJECT	
/SWITCHES:(values)	

Prompt:

FILE? file-spec [,...]

Command Parameter:

file-spec,...	Specifies a COBOL source program to be compiled. If a file-spec does not include a file type. .CBL. is assumed.

Command Qualifiers:

/OBJECT[:object-file-spec] /NOOBJECT	Specifies that an object file be produced and named as indicated by object-file-spec. /OBJECT is the default qualifier. The default file name is the name of the first source file. The default file type is .OBJ. /NOOBJECT specifies that no object file is to be produced.
/LIST[:list-file-spec] /NOLIST	/LIST specifies that the listing be produced and named according to list-file-spec. The default file name is the name of the source file. The default extension is .LST. /NOLIST specifies that no listing file be produced. This is the listing default qualifier.
/SWITCHES: (/values)	Passes optional switches directly to the compiler in keyword form. The switch values are:

ERR:n	CREF
ACC:n	SYM:n
MAP	NORUN
LOD	RUN
CVF	HELP
USW:n. . . :m	TST

Each switch value must be preceded by a slash. For details regarding these switches, see the TRAX COBOL Language User's Guide.

Example:

This command compiles the source file COBPRG. CBL. The object file name defaults to COBPRG. OBJ, and the file OUT. LST contains the listing.

```
>COBOL/LIST:OUT.LST COBPRG.CBL
```

9.7 COPY

The COPY command performs any of these functions, depending on the qualifiers used.

1. Copies a single file such that the output file has the same organization as the input file.
2. Creates a sequential file consisting of a concatenated set of sequential files.
3. Copies a set of files to a corresponding set of files. This is called parallel copying.
4. Creates a sequential file copy consisting of the records from a single sequential, indexed, or relative organized file.

Format:

1. For single or parallel file copying:

COPY [/qualifiers] input-file-spec [/file-qualifier] output-file-spec

Command Qualifiers:

/CONTIGUOUS
/BLOCKSIZE:n
/OWN

File Qualifiers:

/SEQUENTIAL
/RELATIVE
/INDEXED [/KEY:number:n]

2. For concatenating sequential files into one file:

COPY [/qualifiers] (input-file-spec [/SEQUENTIAL] [, . . .])
output-file-spec

Command Qualifiers:

/CONTIGUOUS
/BLOCKSIZE:n
/OWN

Prompts:

```
FROM? input-file-spec [/file-qualified] [....]
TO?   output-file-spec
```

Command Parameters:

input-file-spec	The file specification of the input file. Each file specification must include a file name and a file type.
output-file-spec	The file specification of the output file.

Command Qualifiers:

/CONTIGUOUS	Specifies a contiguous output file. If the /CONTIGUOUS qualifier is omitted, the output file is not necessarily contiguous.
/BLOCKSIZE:n	Specifies a blocksize to be used when copying files to and from magnetic tape.
/OWN	Specifies that the owner of the output file will be the UFD under which it resides.

File Qualifiers:

/SEQUENTIAL	Specifies that the input file has sequential organization.
/RELATIVE	Specifies that the input file has relative organization.
/INDEXED	Specifies that the input file has indexed organization.
/KEY:NUMBER:n	Optionally specifies the record access key for an indexed file. If n=1, it calls for access on the primary key defined for the input file. If n=2, it specifies access on the first alternate key; n=3 specifies the second alternate key, and so on, up to the number of keys defined for the input file.

Notes:

1. If copying disk-to-disk to obtain a sequential output file from an /INDEXED or /REL–ATIVE input file, you must indicate the organization of the input file by means of a file qualifier. If you omit the /RELATIVE or /INDEXED qualifier, the output file organization is the same as the input file. When copying files to magtape, you must specify the organization of the input file.

2. Wildcards are allowed for sequential input files when producing a single, sequential output file. When wildcards appear in the input-file-spec but not in the output-file-spec, the input files are concatenated in the output file. Order of copying depends solely on the order of their appearance in the directory.

3. Wildcards are allowed in the output-file-spec when the directories of the input-file-spec and the output-file-spec are different. Both the file name and file type components of the output file must be represented as wildcards. In this case, each input file is copied into a separate output file with identical file name and file type.

Example:

1. Copy the file RANDOM.DAT from the directory [350, 230] into the current default directory. The copy of the file is unchanged.

```
>COPY [350,230]RANDOM.DAT *.*
```

2. Copy all files with the file type .CBL from the current directory to the directory [40, 41].

```
>COPY *.CBL [40,41]
```

This operation requires write permission in directory [40,41].

9.8 CREATE

The CREATE command creates an empty file. If the file has sequential organization, you may enter text into it as follows:

1. In interactive mode, you enter the formatted command and then type RETURN. On succeeding lines, type the data that you want to place in the file. Type CTRL/Z to indicate the end of the data.

2. In batch mode, the text to be placed in the file follows the command. Any batch command terminates the data file unless the CREATE command includes the qualifier /DOLLARS, in which case only the batch command $EOD can terminate the data.

Files specified as organized /RELATIVE or /INDEXED cannot accept data at the time of creation.

Format:

CREATE [/qualifiers] file-spec

Command Qualifiers:	*Default:*
/ALLOCATION:n	n-0
/BUCKETSIZE:m	m-1
/CONTIGUOUS	
/DOLLARS	
/FORMAT:record-type	VARIABLE=0
FIXED:n	
VARIABLE[:n]	
CONTROLLED[:n]	
/PROCTECTION:code	[RWED, RWED, RWED, R]
/RELATIVE	
/SEQUENTIAL	/SEQUENTIAL
/INDEXED/KEY:value	

Prompt:

FILE? file-spec

Command Parameter:

file-spec	The file specification for the new file must include a file name and a file type.

Command Qualifiers:

/ALLOCATION:n	Specifies n blocks of initial allocation for the file.
/BUCKETSIZE:m	Allowed only with indexed and relative files; specifies a unit of allocation of m blocks for each bucket. In TRAX, m may be a maximum of 16.
/CONTIGUOUS	Specifies contiguous space allocation for the file.
/DOLLARS	Specifies that the data to be entered into the created file contains dollar signs in record position 1. The data entered must be terminated with a $EOD command.
/FORMAT:record type	Specifies the record type of the file.

The following record types are available:

FIXED:n	Specifies fixed length records n bytes long; n is required.

	VARIABLE [:n]	Specifies variable length records. The n parameter defines the maximum length of the record; it is required if /RELATIVE is specified but is otherwise optional.
	CONTROLLED [:n]	Specifies variable length records with a fixed control field. The n variable defines the maximum length of the record, including the fixed control field; it is required if /RELATIVE is specified but is otherwise optional. In all instances, the size of the fixed control field defaults to 2 bytes.

/PROTECTION:code — Protects the file specified in the code parameter. See Section 3.3.2 for a detailed description of the protection code.

/RELATIVE — Specifies relative organization for the file.

/SEQUENTIAL — Specifies sequential organization. This is the default.

/INDEXED — Specifies indexed organization. A /KEY qualifier is also required if /INDEXED is used.

/KEY:value — Specifies the access information for an indexed file. The value parameter may contain the following:

NUMBER:n	Specifies the level of the key field. If n=1, it indicates a primary key, n=2 indicates the first alternate, and so forth.
POSITION:n	Specifies the starting character of the key field.
SIZE:n	Specifies length of the key field.

(NUMBER, POSITION, and SIZE are required for each key value.)

UPDATE	Specifies that the key field is subject to change during the update process.
NOUPDATE	Converse of UPDATE, required on primary keys.
DUPLICATE	Specifies that the record may include duplicate keys. This is implicit if UPDATE is specified.
NODUPLICATE	Converse of DUPLICATE. Illegal with UPDATE.

Table 9-1 shows the legal combinations of UPDATE and DUPLICATE with primary and alternate keys.

Table 9-1 Valid Key Parameter Combinations

KEY TYPE	UPDATE DUPLICATE	UPDATE NODUPLICATE	NOUPDATE DUPLICATE	NOUPDATE NODUPLICATE
Primary	Error	Error	Not supported	Default
Alternate	Default	Error	Allowed	Allowed

Notes:

1. The file-spec must contain a file name and a file type. If an existing version number is not specified, the highest existing version number plus one is used.

2. If sequential organization is specified or defaulted, you can include text in the file as follows:

Interactive	Batch File
>CREATE	$CREATE/DOLLARS file-spec
FILE?	file-spec
contents of file	contents of file, possibly including dollar signs.
CTRL/Z	$EOD

3. If indexed or relative organization is specified, no data is accepted to fill the file.

4. If /INDEXED is specified, a primary key is also required. If any other organization is specified, /KEY is not permitted.

5. The qualifiers /ALLOCATE, /CONTIGUOUS, and /PROTECTION are only applicable when creating an.empty file and not when filling the file with data.

6. The code option /PROTECTION specifies up to four categories of protection: SYSTEM, OWNER, GROUP, and WORLD. Up to four types of access can be specified for each category: READ (R), WRITE (W), EXTEND (E), and DELETE (D). The order of the access type codes R, W, E, and D is fixed. For example:

```
/PROTECTION:(SY:RWED,OWNER:RWED,GROUP:RE)
```

   See Section 3.3.2 for a detailed explanation.

Examples:

This example creates the file ABC.TXT. and accepts lines from the terminal until you type CTRL/Z.

```
>CREATE
FILE? ABC.TXT
THIS IS THE CONTENT OF THE NEWLY-CREATED FILE.
^Z

>
```

This example creates file ACCOUNT.NDX as an indexed file with one key of reference which appears in the first byte of each record and is 10 bytes long.

```
>CREATE/INDEXED/KEY:(NUMBER:1,SIZE:10,POSITION:1)
FILE? ACCOUNT.NDX
>
```

In this example of batch usage, the $CREATE command creates a file that contains batch commands. The file created, COMMAND.CMD, begins with a $COBOL command and ends with a $RUN command. Since the records to be placed in COMMANDS.CMD contain dollar signs in record position 1, the /DOLLARS qualifier is necessary on $CREATE to identify all information up to the $EOD command as data.

```
$JOB
$CREATE/DOLLARS COMMANDS.CMD
$COBOL A.CBL
$LINK A.OBJ,[1,1]COBLIB/LIB,[1,1]RMSLIB/LIB
$RUN A
$EOD
$COPY MYFILE.CBL NEWFILE.CBL
$EOJ
```

9.9 CREATE/DIRECTORY

The CREATE/DIRECTORY command creates a User File Directory (UFD) on the specified disk and enters its name into the Master File Directory (MFD) on the disk. The volume must be initialized and mounted before the CREATE/DIRECTORY command. You can create a UFD only on your private allocated device.

Format:

CREATE/DIRECTORY [/qualifier] [device-name] ufd

Command Qualifiers:	*Default:*
/ALLOCATION:n	n=32
/PROTECTION:code	[RWED, RWED, RWED, R]
/VOLUME_LABEL:volume-id	

Prompt:

DEVICE AND/OR DIRECTORY? device-name ufd

Command Parameters:

device-name

The device name of the disk device on which the directory is to be created. A list of device names is provided in Section 3.2.1. Magnetic tape volumes do not contain directories.

When no device name is specified, the User File Directory is created on the current default system disk.

ufd

The User File Directory to be created. This parameter is required.

The ufd is in the following format:

[ggg, mmm]

The group number, ggg, and the member number, mmm, are octal values from 0 to 377. The brackets are required, with no space between the device name and the left bracket.

Command Qualifiers:

/ALLOCATION:n

Initially allocates "n" directory entries (rounded up to the next multiple of 32). The value n is a decimal number.

/PROTECTION:code	Establishes the access rights for the directory file. The contents of the protection code field are described in Section 3.2.2.
/VOLUME_LABEL:volume-id	The volume-id is a name associated with each volume to verify the correct volume is used. When the incorrect volume-id is specified, the command is ignored. The volume-id consists of an alphanumeric string, 1 to 12 characters long.

Example:

This example creates the User File Directory [200, 34] on DB1:.

```
>CREATE/DIRECTORY DB1:[200,34]
```

9.10 $DATA

The $DATA command indicates the beginning of a batch data block. A data block is necessary whenever you must supply data to a task running under batch control.

In a batch file, any line that does not begin with a dollar sign is treated as data. Thus in many cases you can omit the $DATA command.

The $DATA command is required only when you need to use one of its qualifiers, as described below.

Format:

$DATA [/qualifier]

Command Qualifiers:

/DOLLARS
/NOCOPY

Prompt: None.

Command Parameters: None.

Command Qualifiers:

/DOLLARS	Alerts the system that lines of data may begin with dollar signs. Without this qualifier, lines that begin with dollar signs are treated as commands and thus terminate the data block. When this qualifier is present, all information is treated as data until an $EOD command is encountered.

/NOCOPY	Specifies that the data block to follow not be included in the log file for the batch job.

Note:
In general, you need preface a data block with a $DATA command only if you need to use the /DOLLARS qualifier, the /NOCOPY qualifier, or both.

Example:

The batch job includes a $RUN command that requires input data. The data includes some lines that begin with dollar signs. The data block is not included in the Log File.

```
$JOB
$RUN PROCESS
$DATA/NOCOPY/DOLLARS
INCOME
$76.05
$346.55
$5.80
SPENT
$84.00
$4.89
$EOD
$EOJ
```

9.11 DEALLOCATE

The DEALLOCATE command releases a private device, permitting other users to access the device. You can deallocate only your private devices. The system automatically deallocates private devices when its owner logs off the system.

Format:

DEALLOCATE device-name

Prompt:

DEVICE? device-name

Command Parameter:

device-name	The device name of the device to be deallocated. A list of device names is provided in Section 3.2.1.

Command Qualifier: None.

Example:

This example deallocates the disk DB2:. The system and other private users are now permitted to allocate DB2:.

```
>DEALLOCATE DB2:
```

9.12 DEASSIGN

The DEASSIGN command deletes the specified logical device name from the system logical name table. Nonprivileged users can deassign logical names only at the local level.

Format:

>DEASSIGN [/qualifier] [logical name]

Command Qualifier:	*Default:*
/LOCAL	/LOCAL

Prompt:

LOGICAL NAME?	logical-name

Command Parameter:

logical-name	Specifies a logical device name to be removed from its current device assignment.

Command Qualifier:

/LOCAL	Causes local deassignment. This is the default.

Example:

This command deletes the logical device name US1: from the system at the local level.

```
>DEASSIGN
LOGICAL NAME? US1:
```

9.13 DELETE

The DELETE command has two functions:

- Deletes one or more specified files from the Directory.
- Deletes a specified job from a specified queue.

Each form of the command has its own particular format and rules, as described in the ensuing subsections. See also the description of the PURGE command.

9.13.1 DELETE File

This form of the DELETE command deletes the specified file(s) from the Directory and releases the occupied space. Before deleting any file, you must have delete access to the file. See the DIRECTORY/FULL command to display file access rights.

Format:

DELETE file-spec [, . . .]

Prompt:

FILE? file-spec [, . . .]

Command Parameter:

file-spec	The file specification of the file to be deleted. Each file specification must include a file name, file type, and version number.

Command Qualifier: None.

Notes:

1. Wildcards are allowed in the directory, file name, file type, or file version components of the file specification.
2. You may delete only those files for which you have delete (D) access rights.

Example:

This example deletes all versions of TEST. TSK;*

```
>DELETE TEXT.TSK;*
```

9.13.2 DELETE Queued Job

This form of the DELETE command deletes a job entry and its associated file entries from a specified queue. The files themselves are not deleted.

Format:

DELETE/QUEUE queue-id

Prompt:

QUEUE NAME? queue-id

Command Parameter:

queue-id

Specifies a queued job in one of two forms:

queue name/JOB: [uic] jobname

or

ENTRY: (m, n)

If you select the first, you must include the qualifier: /JOB: [uic] jobname. If [uic] is not specified, your own UIC is the default. Jobname specifies a job currently in the queue.

The following form specifies an internal job entry identifier:

ENTRY: (m, n)

The value (m, n) specifies an internal identifier assigned by the system when the job is originated. You can learn this identifier by using the SHOW QUEUE command to obtain a FULL listing. The system queue identifier ENTRY must not be abbreviated.

Command Qualifier:

/QUEUE

Specifies that a job entry will be deleted from a queue; this qualifier is required to identify the DELETE/QUEUE function.

Parameter Qualifier:

/JOB: [uic] jobname

Specifies the job to be deleted when the queue-name parameter is selected.

Note:
If the job is being processed when the DELETE/QUEUE command is entered, the job is aborted and then deleted from the queue.

9.14 DIRECTORY

The DIRECTORY command displays the directory information of specified files or the contents of your current default directory.

Format:

>DIRECTORY [/qualifiers] [file-spec [, . . .]]

Command Qualifiers:	*Default:*
/FULL	/BRIEF
/BRIEF	
/SUMMARY	
/FREE	
/PRINT	
/OUTPUT:file-spec	
/ATTRIBUTES	

Command Parameter:

file-spec	Specifies the directory entries to be listed. If omitted, all directory entries are listed.

Command Qualifiers:

/BRIEF	The entry display contains only the file specification, block count, and creation date. This is the default. See Example 1.
/FULL	A complete directory entry listing is displayed. See Example 2.
/SUMMARY	Only the total number of blocks allocated to all files in the directory is displayed.
/FREE	The free space available either on the system device or the specified device is displayed.
/ATTRIBUTES	Gives a description of the specified files that includes the full RMS attributes. See Example 3.
/OUTPUT:file-spec	Forces the display to be placed in a file according to the file-specification.
/PRINT	Causes the display to appear on the line printer.

Notes:

1. If no files are specified, a directory list of your current default UFD on your default device is given.

2. One or more file-specs may be given.

3. If no filetype is specified, a wildcard for filetype is assumed. If no file version is given, the highest version number is assumed.

Example:

The following examples of DIRECTORY commands show the type of information you can expect in response to different qualifiers. The same file specification is used in each case.

1. This is a /BRIEF (default) directory listing.

```
DIRECTORY DB0:[40,40]
18-JUL-78 18:25

A.LST;1              1.          18-JUL-78 14:14
A.ODL;1              1.          18-JUL-78 14:14
A.OBJ;1              2.          18-JUL-78 14:14
A.TSK;1              27.      C  18-JUL-78 14:14
A.CBL;2              1.          18-JUL-78 14:15

TOTAL OF 32./32. BLOCKS IN 5. FILES
```

2. This is a /FULL directory listing.

```
>DIRECTORY/FULL A.*


DIRECTORY DB0:[40,40]
18-JUL-78 18:26

A.LST;1              (14271,6)       1./1.          18-JUL-78 14:14
  [40,40]  [RWED,RWED,RWED,R]
A.ODL;1              (15040,23)      1./1.          18-JUL-78 14:14
  [40,40]  [RWED,RWED,RWED,R]
A.OBJ;1              (15233,56)      2./2.          18-JUL-78 14:14
  [40,40]  [RWED,RWED,RWED,R]
A.TSK;1              (15432,7)       27./27.    C   18-JUL-78 14:14
  [40,40]  [RWED,RWED,RWED,R]
A.CBL;2              (17211,10)      1./1.          18-JUL-78 14:15
  [40,40]  [RWED,RWED,RWED,R]

TOTAL OF 32./32. BLOCKS IN 5. FILES
```

3. This is an /ATTRIBUTES directory listing.

```
>DIRECTORY/ATTRIBUTES A.*

SY0:[40,40]A.LST;1      FILE ORGANIZATION:      SEQUENTIAL
CREATED: 18-JUL-1978 14:14
FILE PROTECTION:        [RWED,RWED,RWED,R]
RECORD FORMAT:          VARIABLE
RECORD ATTRIBUTES:      CARRIAGE RETURN
FILE ATTRIBUTES:
        ALLOCATION= 1   EXTEND QUANTITY=0

SY0:[40,40]A.ODL;1      FILE ORGANIZATION:      SEQUENTIAL
CREATED: 18-JUL-1978 14:14
FILE PROTECTION:        [RWED,RWED,RWED,R]
RECORD FORMAT:          VARIABLE
RECORD ATTRIBUTES:      CARRIAGE RETURN
FILE ATTRIBUTES:
        ALLOCATION= 1   EXTEND QUANTITY=0

SY0:[40,40]A.OBJ;1      FILE ORGANIZATION:      SEQUENTIAL
CREATED: 18-JUL-1978 14:14
FILE PROTECTION:        [RWED,RWED,RWED,R]
RECORD FORMAT:          VARIABLE
RECORD ATTRIBUTES:
FILE ATTRIBUTES:
        ALLOCATION= 2   EXTEND QUANTITY=0

SY0:[40,40]A.TSK;1      FILE ORGANIZATION:      SEQUENTIAL
CREATED: 18-JUL-1978 14:14
FILE PROTECTION:        [RWED,RWED,RWED,R]
RECORD FORMAT:          FIXED=512
RECORD ATTRIBUTES:
FILE ATTRIBUTES:
        ALLOCATION= 27  EXTEND QUANTITY=0
                        CONTIGUOUS

SY0:[40,40]A.CBL;2      FILE ORGANIZATION:      SEQUENTIAL
CREATED: 18-JUL-1978 14:15
FILE PROTECTION:        [RWED,RWED,RWED,R]
RECORD FORMAT:          VARIABLE
RECORD ATTRIBUTES:      CARRIAGE RETURN
FILE ATTRIBUTES:
        ALLOCATION= 1   EXTEND QUANTITY=0

>
```

9.15 DISMOUNT

The DISMOUNT command logically disconnects the specified volume from the system.

Format:

DISMOUNT device-name [volume-label]

Prompts:

DEVICE? device-name
VOLUME LABEL? [volume-label]

Command Parameters:

device-name	The specification of the device containing the volume to be dismounted.
volume-label	An optional parameter, and if present, dismount operation is executed only if the label matches the volume label. This parameter is required for magnetic tape volumes.

Command Qualifiers: None.

Notes:

1. As with the MOUNT command, DISMOUNT may take a logical device name in place of a physical device name.

2. Volume label is mandatory for magnetic tape volumes and optional for all other volumes. When volume-label is specified, the system verifies that the correct volume is used. The command is rejected when an incorrect volume label is specified. Volume label may be up to 6 alphanumeric characters for magnetic tape volumes, and up to 12 alphanumeric characters for other volumes.

Example:

This example dismounts the disk DB0 and verifies that its volume label is VOLNAME.

```
>DISMOUNT DB0:  VOLNAME
```

9.16 EDIT

The EDIT command invokes the editor to edit or create the specified file.

Format:

EDIT file-spec

Prompt:

FILE? file-spec

Command Parameters:

file-spec	The specification of the file to be edited. The file specification must include a file name and a file type.

Command Qualifiers: None

Notes:

1. If you do not provide a version number, the highest exisiting version is used. If a file does not exist as specified, a new file is created with version number 1.

2. Details of the use of the editor may be found in the DEC EDITOR Reference Manual.

Example:

The following sequence initiates an edit session on the file EASY.CBL.

```
>EDIT
FILE? EASY.CBL
*
```

*The asterisk is a prompt for an editor command. When you want to terminate the edit session, enter the editor command EXIT. The DCL prompt (>) will appear on the terminal.

9.17 $EOD

The $EOD (End of Data) command terminates a data stream initiated by a $DATA command, or the input to a file created by a $CREATE/DOLLARS command. The command may only be given in batch mode. A data stream that is not initiated by a $DATA command does not require an $EOD command for termination. See the section of Chapter 6 entitiled, "Batch Data Blocks."

Format:

$EOD

Note: The command has no parameters or qualifiers.

Example:

This example uses $EOD to terminate a data block. The /DOLLARS qualifier instructs the system to accept the following lines of text as input to the file rather than batch commands to be processed.

```
$CREATE/DOLLARS TRAN.DAT
#UPDATE DATA FOR 27FEB
A601-450
$35.42
$102.99
T79-132
$824.09
$EOD
```

9.18 $EOJ

The $EOJ (End of Job) command terminates a batch job, dismounting and deallocating any allocated devices.

Format:

$EOJ

Notes:

1. The command has no parameters or qualifiers.

2. The $EOJ command is the last command in a batch job command stream. An $EOJ command is implied at the end of a batch command file if one is not included explicitly.

Example:

The $EOJ command ends the batch job and is analogous to a LOGOUT command ending an interactive terminal session.

```
$JOB
$MOUNT DB0: MAR27A
$RUN TEST
$DISMOUNT DB0:
$EOJ
```

9.19 $GOTO

The $GOTO command is used only in batch mode. $GOTO suppresses execution of all commands up to the first command that is prefixed by a specified label.

Format:

$GOTO label

Prompts: None.

Command Parameter:

LABEL	Is an alphanumeric string that must also appear, together with a colon, at the beginning of a later command.

Command Qualifier: None.

Notes:

1. $GOTO can be used by itself or as an action in an ON or IF command.

2. When control is transferred, the system scans the file forward, ignoring commands until it finds a command with a label that matches the $GOTO parameter. If no matching label is found, no further processing takes place within the batch command file.

3. $GOTO cannot transfer control to an earlier labeled command.

Example:

In this example, the linking and running of MY PROG is halted if an error occurs at any point, and the task OLDPROG is run instead. OLDPROG is not run if no error occurs. See the description of the $ON command for further clarification.

```
$JOB SYSTEM
$ON ERROR THEN GOTO L10
$LINK/BASIC MYPROG
$RUN MYPROG
$GOTO L20
$L10:  RUN OLDPROG
$L20:  RUN TEST
$EOJ
```

9.20 HELP

The HELP command displays information about the commands and their associated qualifiers. When no parameters are specified, the system displays a complete list of commands on the requesting terminal. When a qualifier or other keyword is displayed with two asterisks (**) in the HELP output, further information is available on the keyword.

Format:

HELP [command-name [keyword]]

Prompt: None.

Command Parameters:

command-name	The command and its qualifiers (if any) are displayed. When a qualifier is displayed with two asterisks (**) after it, there is further information available on the qualifier.
keyword	Displays the keywords available for the qualifier as applied to the command.

Command Qualifier: None.

Examples:

The following examples show the use of HELP with and without a second parameter.

```
>HELP UNLOCK
        UNLOCK FILESPEC[,FILESPEC(S)]

>HELP SHOW QUEUE
        SHOW QUEUE QUEUENAME     OPTION(S)
                   ENTRY:(N,N)   JOB:[[UIC]]JOBNAME
                   ALL           USER:[G,M]
                                 NUMBER
                                 ALL
                                 BATCH
                                 PRINT
                                 PRIORITY:N
                                 FORMS:N
                                 FULL
                                 BRIEF
```

9.21 $IF

The $IF command specifies alternative action if a specified status condition occurs on a command. It is used only in batch jobs.

Format:

$IF status-level THEN action

Prompts: None.

Command Parameters:

status-level	One of the following: SUCCESS WARNING ERROR SEVERE ERROR
action	One of the following: GOTO label CONTINUE STOP

Command Qualifiers: None.

Notes:

1. The status-level resulting from the execution of the command preceding the $IF command is checked. If that status-level is equal to the status-level given in the $IF command, the THEN clause is executed. Otherwise the THEN clause is not executed, and the batch job continues with the next command in the file.

2. The label of the $GOTO phrase must be the label of a command appearing after the $IF command.

Example:

The following $IF command causes the batch job to terminate immediately if the preceding command results in a status of ERROR. Otherwise, the job proceeds sequentially.

```
$IF ERROR THEN STOP
```

See Section 6.7.5 for this example.

9.22 INITIALIZE

The INITIALIZE command produces a file-structured volume on disk or magnetic tape. The command destroys all existing files on the volume. The system creates a Master File Directory (MFD) on the disk or creates a volume label and dummy file on the magnetic tape.

Format:

INITIALIZE device-name volume label

Command Qualifiers:	*Default:*
/DENSITY:n	n=800
/ENTENSION:n	n=5
/HEADERS:n	
/INDEX:location	location=MIDDLE
/MAXIMUM:p	varies with disk type
/OWNER: [ggg, mmm]	[1, 1]
/PROTECTION:code	[RWED, RWED, RWED, R]
/[NO] VERIFIED	/VERIFIED
/VOLUME PROTECTION:code	[RWED, RWED, RWED, R]
/WINDOW:a	a=7

Prompts:

DEVICE? device-name
VOLUME LABEL? volume-label

Command Parameters:

device-name

The device name of the device to be initialized. Device-name can be a physical device name or an assigned logical device name.

volume-label

The volume-label is a name associated with each volume. The volume-label may be up to 6 alphanumeric characters for magnetic tape volumes and up to 12 alphanumeric characters for all other volumes.

The volume-label is requested by other commands, such as the MOUNT command, to ensure the proper volume is used.

Command Qualifiers:

/DENSITY:n

Specifies the recording density in bits per inch (bpi) of the magnetic tape to be initialized. Acceptable values for n are either 800 bpi or 1600 bpi. When not specified, 800 bpi is used.

/EXTENSION:n

Specifies the default number of blocks a disk file shall be extended, when it exhausts its current space allotment. The value of n is decimal.

/HEADERS:n

Specifies the initial number of allocated file headers in the index file. The value of n is decimal.

/INDEX:location

Positions the index file, on the volume, at the specified location. Possible location values are:

BEGINNING	Place the index file at the beginning of the volume.
MIDDLE	Place the index file at the middle of the volume.
END	Place the index file at the end of the volume.
BLOCK:n	Place the index file at the "n" block of the volume.

/MAXIMUM:p

Specifies the maximum number of files that the disk volume can contain.

/OWNER: [ggg, mmm]

Specifies the UIC of the owner of the volume. The group number, ggg, and the member number, mmm, are OCTAL values from 0 to 377. The square brackets, [ ], are required syntax.

/PROTECTION:code

Specifies the default protection code that will be applied to files when they are created on the volume. See Section 3.3.2 for description of protection code.

/VERIFIED
/NOVERIFIED

Includes bad block processing in the volume initialization. When specifying VERIFIED, the system reads the bad block file created by the support environment utility, BAD.

When specifying NOVERIFIED, the system accepts block specifications from the terminal. The program prompts for bad blocks with the display:

```
INI>BAD:
```

Bad blocks may be entered in two formats:

nnnnn	A single block. nnnnn specifies an octal disk block number.
nnnnn, mmmm	A contiguous series of mmmm blocks beginning at nnnnn.

A null line (carriage return) terminates bad block input.

/VOLUME PROTECTION:code	Specifies task access rights to the volume. The code format is the same as for the /PROTECTION=code qualifier.
/WINDOW:a	Specifies the decimal number of mapping pointers to be allocated for file windows.

Notes:

1. The /DENSITY qualifier applies only to magnetic tape volumes.
2. The /EXTENSION, /HEADERS, /INDEX, /MAXIMUM, /OWNER, /[NO] VERIFIED, and /WINDOW qualifiers apply to disk volumes only.
3. The /VOLUME PROTECTION qualifier applies to all types of volumes.
4. You can only initialize volumes mounted on your private device; that is, those devices for which you have issued an ALLOCATE command.

Examples:

This example initializes the disk, DB0: , with the volume-id of VOLLABEL.

```
>INITIALIZE DB0: VOLLABEL
```

This example initializes DCLVOL2 on DB1: with the index file located at the end of the volume. The owner UIC is [40, 40].

```
>INITIALIZE/INDEX:END/OWNER:[40,40] DB1: DCLVOL2
```

9.23 $JOB

The $JOB command is used only in batch mode and marks the beginning of a batch job. It is the batch mode equivalent of an interactive LOGIN command.

Format:

$JOB [/qualifier] jobname [uic]

Command Qualifier:	*Default:*
/TIME=xx	No time limit

Prompts: None.

Command Parameters:

job-name	Specifies the name by which the batch job will be identified in the batch log.

uic	Specifies the User Identification Code. This is a privileged parameter that enables the batch job to log in under a different account than the one from which the job was submitted.

Command Qualifier:

/TIME=xx	Specifies the maximum number of minutes in wall clock time that the batch job is allowed to run. This parameter is optional; if omitted, the system assumes no time limit.

9.24 LIBRARIAN

The Librarian command allows you to create, delete, and maintain object module libraries and MACRO-11 source libraries.

Format:

```
>LIBRARIAN operation
```

Prompt:

```
OPERATION? operation
```

Command Parameter:

Operation	Specifies the librarian operation to be performed. It consists of one of the following keywords, followed by a set of associate parameters and qualifiers appropriate for that keyword.

CREATE	LIST
DELETE	REPLACE
EXTRACT	SQUEEZE
INSERT	

The keyword is considered an extension of the command name.

The following subsections describe each of these operations.

9.24.1 LIBRARIAN CREATE

The LIBRARIAN CREATE command creates, and optionally populates, a library file.

Format:

```
LIBRARIAN CREATE [/qualifiers] lib-spec [input-file-spec]. . .
```

Command Qualifiers:	*Default:*
/SIZE:n	n=100
/EPT:n	See Qualifier description
/MNT:n	n=256
/TYPE:OBJECT	/TYPE=OBJECT
MACRO	
/SELECT_SYMBOLS	
/SQUEEZE	
/NOENTRY POINTS	

Prompts:

LIBRARY? lib-spec
MODULES? [input-file-spec],...

Command Parameters:

lib-spec	Specifies the name of the library file to be created. If no file type is given, the default is .OLB for object module libraries and .MLB for MACRO module libraries.
input-file-spec	Optionally specifies one or more files that will constitute input to the new library file. If the parameter is not present, an empty library file is created. Files specified in this parameter are called library modules.

Command Qualifiers:

/SIZE:n	Specifies the size of the library file in 512 byte blocks. Default is 100.
/EPT:n	Specified the number of entry points to allocate in the entry point table (EPT). The default value is 512 for object libraries but the number of entry points for MACRO libraries is always 0.
/MNT:n	Specifies the number of entries to allocate in the module name table. It must not exceed 4096 and is rounded up to the nearest multiple of 64. The default value is 256.
/TYPE:library-type	Defines the type of libraries to be created as either OBJECT or MACRO. The default is OBJECT.

/SELECT_SYMBOLS	Specifies that the LINK command will use the created library to define required global symbols at task build time (for object files only).
/SQUEEZE	Specifies that the MACRO file should be reduced by erasing all trailing blanks and tabs, blank lines, and comments from the source text (for MACRO library files only).
/NOENTRY POINTS	Specifies that the modules specified in input-file-spec are inserted into the library in lib-spec, but the entry points in the modules are not entered in the entry point table (EPT).

Note:
If the qualifiers /SELECT and /SQUEEZE are used with CREATE, the input-file-spec parameter must appear.

Example:

This example creates an object library file MYLIB.OLB with 100 blocks default size and 512 entry point and 256 module name entries containing the two object modules, OBJ1.OBJ and OBJ2.OBJ.

```
>LIBRARIAN
OPERATION? CREATE/TYPE:OBJECT
LIBRARY? MYLIB.OLB
MODULES? OBJ1.OBJ,OBJ2.OBJ
```

9.24.2 LIBRARY DELETE

The LIBRARY DELETE command performs two types of deletions.

- Deletes modules and all their associated entry points from the specified library file.
- Deletes specified entries in the entry point table (EPT).

Format:

LIBRARIAN DELETE [/qualifiers] lib-spec entry-name, . . .

Command Qualifiers:	*Default:*
/MODULES	/MODULES
/GLOBAL_SYMBOLS	

Prompts:

```
LIBRARY? lib-spec
ENTRIES? entry-name, . . .
```

Command Parameters:

lib-spec	Specifies the library file that contains the modules or entries to be deleted. If a file type is not expected, the default is .OLB.
entry-name	Specifies the module name(s) or entry name(s).

Command Qualifiers:

/MODULES	Deletes the specified modules and is the default qualifier.
/GLOBAL_SYMBOLS	Deletes the specified EPT entries.

Note:
Up to 15 modules may be deleted in one DELETE operation. A deleted module is marked as deleted but remains physically in the file until a SQUEEZE operation is performed.

Example:

This example deletes the object module NAMEA from the object library MYLIB.OLB.

```
>LIBRARIAN
OPERATION? DELETE/MODULES
LIBRARY? MYLIB.OLB
ENTRIES? NAMEA
```

9.24.3 LIBRARIAN EXTRACT

The LIBRARIAN EXTRACT command enables the extraction of defined modules from a specified library and concatenates them in a specified file.

Format:

LIBRARIAN EXTRACT/OUTPUT:file-spec lib-spec modules-spec, . . .

Prompts:

OPERATION? EXTRACT/OUTPUT:file-spec
LIBRARY? lib-spec
MODULES? module-spec

Command Parameters:

file-spec	Specifies the file that is to receive the extracted modules. If a file type is omitted, the default file type is .MAC—if the library is a MACRO library—and OBJ if the library is an object library.

lib-spec	Specifies the library that contained the modules to be extracted.
module-spec	Defines the module(s) to be extracted.

Command Qualifiers: None.

Example:

This example extracts modules MODULE1.OBJ, MODULE2.OBJ from the library file MYLIB.OLB and concatenates them in file OBJ3.OBJ.

```
>LIBRARIAN
OPERATION? EXTRACT/OUTPUT:OBJ3.OBJ
LIBRARY? MYLIB.OLB
MODULES? MODULE1, MODULE2
```

9.24.4 LIBRARIAN INSERT

The LIBRARIAN INSERT command inserts modules into a specified library file. Any number of input files are allowed.

Format:

LIBRARIAN INSERT [qualifier(s)] lib-spec input-file-spec, . . .

Command Qualifiers:

/SELECT_SYMBOLS
/SQUEEZE
/NOENTRYPOINTS

Prompts:

LIBRARY? lib-spec
FILE? input-file-spec, . . .

Command Parameters:

lib-spec	Specifies the library file into which modules are to be inserted.
input-file-spec	Specifies the object modules to be inserted.

Command Qualifiers:

/SELECT_SYMBOLS	Specifies that the LINK command will use the created library to define required global symbols at link time (for object files only).

/SQUEEZE	Specifies that a MACRO file should be reduced by easing all trailing blanks and tabs, blank lines and comments from the source text (for MACRO files only).
/NOENTRYPOINTS	Specifies that the modules specified in input-file-spec are inserted into the library in lib-spec, but the entry points in the modules are not entered in the entry point table (EPT).

Example:

This example inserts the MACRO file ONE.MAC into the MACRO library MACLIB.MLB, stripping off all unnecessary characters.

```
>LIBRARIAN
OPERATION? INSERT/SQUEEZE
LIBRARY? MACLIB.MLB
FILE? ONE.MAC
```

9.24.5 LIBRARIAN LIST

The LIST operation causes a library file directory to be printed or to be sent to an output file. The former is the default.

Format:

LIBRARIAN LIST [/qualifiers] lib-spec

Command Qualifiers:	*Default:*
/ENTRIES	/ENTRIES
/FULL	
/OUTPUT:list-file-spec	

Prompt:

LIBRARY? lib-spec

Command Parameter:

lib-spec	Specifies the library file to be listed.

Command Qualifiers:

/ENTRIES	Causes a directory of all modules to be listed together with entry points for each. This list is the default.

/FULL	Causes a directory of all modules to be listed giving full module descriptions; size, date of insertion and version.
/OUTPUT:list-file-spec	Causes the output to be sent to the specified file. The default file type is .LST.

Example:

This example lists at the user's terminal a directory of all modules and their full descriptions from the library MYLIB.OLB.

```
>LIBRARIAN LIST/FULL MYLIB.OLB
```

9.24.6 LIBRARIAN REPLACE

The LIBRARIAN REPLACE command replaces a module in the library with a new module of the same name. That is, a new module that has the same name as a module already contained in the library, replaces the existing module. The old module is deleted.

Format:

LIBRARIAN REPLACE [/qualifiers] library-spec module-spec

Command Qualifiers:

/SELECT_SYMBOL
/SQUEEZE
/NOENTRYPOINTS

Prompts:

LIBRARY? lib-spec
FILE? module-spec

Command Parameters:

library-spec	Specifies the library file containing the module to be replaced.
module-spec-list	Specifies one or more files containing the new modules.

Command Qualifiers:

/SELECT_SYMBOL	Specifies that the LINK command will use the created library to define required global symbols at link time (for object files only).

/SQUEEZE	Specifies that the size of a MACRO file should be reduced by erasing all trailing blanks and tabs, blank lines, and comments from the source text (for MACRO files only).
/NOENTRYPOINTS	Specifies that the modules specified in module-spec-list are inserted into the library in lib-spec, but the entry points in the modules are not entered in the entry point table (EPT).

Example:

This example replaces the module in MACLIB.MLB with the same name as NEWMOD.MAC.

```
>LIBRARIAN
OPERATION? REPLACE
LIBRARY? MACLIB.MLB
FILE? NEWMOD.MAC
```

9.24.7 LIBRARIAN SQUEEZE

The LIBRARIAN SQUEEZE command creates a new library file consisting of all modules from the old file that have not been logically removed by LIBRARY DELETE and LIBRARY REPLACE operations, omitting all modules that have been deleted or replaced (but are still physically present). This creates a compressed version of the library file. The old library file is not automatically deleted after creation of the new file.

Format:

>LIBRARIAN SQUEEZE [/qualifiers] lib-spec [new-lib-spec]

Command Qualifiers:	*Defaults:*
/SIZE:n	/SIZE:100
/EPT:n	EPT:512 for object libraries, 0 for macro libraries
/MNT:n	MNT:256

Prompts:

LIBRARY? lib-spec
NEW LIBRARY? new-lib-spec

NEW LIBRARY is prompted only if LIBRARY is prompted.

Command Parameters:

lib-spec	Specifies the library file to be compressed.

new-lib-spec	Specifies the compressed library file. If omitted, a new version of lib-spec.

Command Qualifiers:

/SIZE:n	Specifies the size in 256 word blocks of the compressed file. Default is 100.
/EPT:n	Specifies the number of entry points to allocate in the entry point table (must not exceed 4096). The default is 512 for object libraries. n is rounded up to the nearest multiple of 64.
/MNT:n	Specifies the number of entries to allocate in the module name table. It must not exceed 4096 and is rounded up to the nearest multiple of 64. The default value is 256.

Example:

This example compresses the library LIB1.OLB to 150 blocks with (by default) 512 EPT entries and 256 MNT entries. The compressed file is renamed LIB2.MLB.

```
>LIBRARIAN
OPERATION? SQUEEZE/SIZE:150
LIBRARY? LIB1.OLB
NEW LIBRARY? LIB2.MLB
```

9.25 LINK

The LINK command invokes the TRAX linker to convert object modules into executable task images. It produces output as directed by command qualifiers. For further information, see the TRAX Linker Reference Manual.

Format:

LINK [/qualifiers] [file-spec [/file-qualifiers], . . .]

Command Qualifiers:	*Default:*
/BASIC	
/CHECKPOINT:SYSTEM :TASK	/CHECKPOINT:S
/NOCHECKPOINT	
/CROSS_REFERENCE	
/DEBUG [:debug-file-spec]	
/[NO] DUMP	/NODUMP

/[NO] FULL_SEARCH	
/MAP: map-file-spec [/FULL] /NARROW /SHORT /WIDE	
/NOMAP	/NOMAP
/OPTIONS[:file-spec]	
/OVERLAY[overlay-file-spec]	
/[NO] RECEIVE	/RECEIVE
/SEQUENTIAL	
/SYMBOLS [symbol-file-spec]	
/[NO] SYMBOLS	/NOSYMBOLS
/TASK: task-file-spec	TASK=default-file-spec
/NOTASK	
File Qualifiers:	*Default:*
/[NO] CONCATENATED	/CONCATENATED
/DEFAULT_LIBRARY: file-spec	
/LIBRARY [:module-list]	
/[NO] MAP	See qualifier description.
/SELECT_SYMBOLS	See qualifier description.

Prompt:

FILE? file-spec [/file-qualifiers], . . .

Command Parameter:

file-spec

Specifies an input file containing object modules. It must not be present if the command qualifier /OVERLAY is specified.

If the file name is given with no file type, the default file type of .OBJ is used for an object file and .OLB for a library. File specifications for symbol table files must include a file type .STB and specifications for an overlay description file must include file type .ODL.

Command Qualifiers:

/BASIC

Identifies the input file as a command file produced by issuing the BUILD command to the BASIC-PLUS-2 compiler. The Linker decodes the command file and the task image file according to information supplied in the command file.

The /BASIC qualifier is valid only if the input file was generated this way.

/CHECKPOINT [:keyword]
/NOCHECKPOINT

Identifies the Linker task as checkpointable when /CHECKPOINT is specified. The optional keyword is SYSTEM or TASK; this specifies where the checkpoint space is allocated. TASK requests checkpoint space within the task image file, and SYSTEM requests system checkpoint space. The qualifier /CHECKPOINT:SYSTEM is the default.

You should avoid specifying /NOCHECKPOINT, as this option seriously degrades overall system performance. See Note for explanation.

/CROSS_REFERENCE

Requests that a symbol cross-reference listing be appended to the memory allocation (MAP) file; thus the /MAP qualifiers must also be present for this qualifier to be effective.

If /CROSS_REFERENCE is not specified, no cross-reference listing is produced.

/DEBUG[:debug-file-spec]

Specifies incorporation of a debugging aid in the task image file. If debug-file-spec is omitted, the system standard debugging aid is used. You can incorporate a different debugging aid by specifying a debug-file-spec. The user generated debugging aid must be in object module format.

See the TRAX Linker Reference Manual for further information including a debugging aid.

/DUMP
/NODUMP

/DUMP requests a post-mortem dump if your task is terminated abnormally.

/NODUMP is the default.

/FULL_SEARCH
/NOFULL_SEARCH

Specifies a full search of all co-tree overlay segments for a matching definition or reference, when processing modules from the default object module listing.

NOFULL_SEARCH is the default.

/MAP[:map-file-spec[/map-file-qualifier]]
/NOMAP

Instructs the linker to produce a memory allocation file (with file type .MAP) when linking the task image file.

If you specify /MAP without a map-file-spec, the memory allocation file is spooled directly to the line printer. It remains on your file directory taking the task file name and the file type .MAP until it is deleted after printing.

If you include the map-file-spec, you may omit the file type field and the linker will use the file type .MAP.

/NOMAP is the default if /MAP is not specified.

The following file qualifiers may be applied to the map-file-spec.

/FULL	The Linker will include all modules in the memory allocation file, even those which explicitly or by default have the NOMAP input file qualifier.
/NARROW	The Linker produces a map listing 72 characters wide, suitable for printing on an output terminal.
/SHORT	Tells the Linker to include only the segment headings in the memory allocation file.
/WIDE	Produces a map 132 characters wide, suitable for printing on a line printer. When /MAP is specified, this is the default file qualifier.

/OPTIONS[:file-spec]

Provides or prompts for Linker option input. See the TRAX Linker Reference Manual for detailed information on Linker options.

If no file-spec argument is present, the Linker prompts for Linker option input lines as follows:

```
OPTIONS?
```

This prompt continues after each line of option input that you enter, until you type a line that ends with a slash (/), as follows:

```
OPTIONS?  /<CR>
```

OPTIONS? : OPTION-input /<cr>

When the file-spec is included, the linker treats that file as a series of option input lines. Interactive prompting for options does not occur. The default file type for the input file is .CMD.

/OVERLAY[:overlay-file-spec]

Specifies an Overlay Description Language (ODL) file that will govern the creation of the task image file.

Only an overlay description file is allowed with this qualifier. See the TRAX Linker Reference Manual for information on overlay descriptions.

/[NO]RECEIVE

Enables the resultant task to receive direct messages via the executive SEND directive.

/RECEIVE is the default. To disable the feature, the /NO RECEIVE qualifier is required.

/SEQUENTIAL

Causes the task image to be constructed from the object files in the order stated in the Link command string. If /SEQUENTIAL is not present in the command string, the Linker records the object program files alphabetically, not sequentially.

See the TRAX Linker Reference Manual for further description of task image storage allocation in detail.

/SYMBOLS[symbol-file-spec]
/NOSYMBOLS

/SYMBOLS specifies creation of a symbol table definition file by the Linker. If symbol-file-spec is present, the file type is optional; if the type is absent, it defaults to .STB.

If symbol-file-spec is absent, the first input file name becomes the file name, with .STB the default file type.

/NOSYMBOLS is the default qualifier.

/TASK [:task-file-spec]
/NOTASK

Specifies the name of the task image file. If task-file-spec is present, the file type is optional; if file type is absent, it defaults to .TSK. If file-spec is absent, the first input file name becomes the file name of the task image file, with .TSK the default file type.

/TASK is the default. If /NOTASK is used, the linker processes the input for unresolved symbol references but does not produce a task image file.

File Qualifiers:

/[NO] CONCATENATED

/CONCATENATED specifies processing of all modules in the input file to form the task image, and is the default.

/NOCONCATENATED causes the Linker to process only the first object module, regardless of the number present.

The /LIBRARY qualifier overrides this qualifier.

/DEFAULT LIBRARY:file-spec

Specifies the default library file to be used for resolving undefined global symbol references. This overrides the default system library LB0=[1,1] SYSLIB.OLB.

If the specified library is empty, the default library reverts to the system library.

/LIBRARY:[(]module[,...)]

Identifies the associated file (that is, the input file specification modified by this qualifier) as an object module library file. /LIBRARY is required for any input library file, and its use is prohibited for any other type of file.

If no modules are specified, the Linker searches the library file to resolve undefined global symbol references. The Linker extracts any and all modules that resolve undefined references and includes them in the task image file.

If you specify module names, the file is defined as a library file (file type .OLB) of relocatable object modules, and the modules named are copied for inclusion in the task image.

The module names are defined at assembly time. You may specify up to eight modules, and only those specified are included in the task image.

To direct the Linker to search a library file for both global symbol references and selected modules needed in the task image, you must include both forms of the qualifier, using separate file specifications.

/[NO] MAP

Specifies inclusion of this file in the memory allocation map.

If /NOMAP is specified, no details of modules contained in the file will appear in the memory allocation map or cross-reference listing. /NOMAP, when qualifying an input file, is overridden by the command qualifier FULL SEARCH.

For a system library file, resident libraries, and common areas, /NOMAP is the default qualifier. For user supplied object module input files, /MAP is the default qualifier.

/SELECT SYMBOLS

Instructs the Linker to search the file only for those global symbols for which an undefined reference exists. The Linker uses only the required symbol definitions.

This qualifier is useful when an input file is the symbol table (file type .STB) output of another LINK command, because it reduces the size of the symbol table search and improves system performance.

If /SELECT SYMBOLS is absent, all global symbols from the input file are included in the task image file; that is the default condition.

If the /LIBRARY or /CONCATENATED qualifier is in effect, /SELECT SYMBOLS is active for each module of the input file.

Note:
Checkpointing is recommended as good programming practice. If a task (called Task A) is running and then a task of higher priority (called Task B) enters the system. Task A can be interrupted if it has been defined as checkpointable. The current state of Task A is recorded in the selected checkpoint area. When its required system resources become available again, it is reinstalled and resumed in the state that existed at the time of the interrupt.

Examples:

The file specification INTEREST .CMD contains a BASIC-PLUS-2 program. The file type .CMD is added to the file name by default.

```
>LINK/BASIC INTEREST
```

After execution of this LINK command, an executable task image called INTEREST.TSK is ready.

The following command directs the task OVERLAY.TSK to be created and the map file OVERLAY.MAP to be generated and spooled. The optional input specifies that DB0: will be assigned to LUN 8. The task will be built from the overlay descriptor file OVERLAY.ODL and will include the standard debugging aid.

```
>LINK/DEBUG/OVERLAY:OVERLAY/OPTIONS/MAP
OPTIONS?ASG=DB0:8
OPTIONS?/
```

9.26 LOGIN

The LOGIN command initiates a user session at a terminal. A valid user-id must be given to ensure that an authorized user is accessing the system. The system records information on CO0: the operator's console, about who is logging onto the system and when the login occurred. When a user created LOGIN.CMD file exists, the system then executes the file from the User File Directory.

The system grants access to the terminal until a LOGOUT command is issued.

Format:

LOGIN user-id password

Prompts:

USERID? user-id
PASSWORD? password

Command Parameters:

user-id

The user-id is either a UIC or the user name associated with the UIC. Valid forms of user-id are:

name
[ggg,mmn]
ggg,mmm
ggg/mmm

The user-id, ggg/mmm, suppresses the login text message after the first time a user logs into the system during any given day.

The group number, ggg, and the member number, mmm, are octal values from 0 to 377.

password	A 1- to 6- character alphanumeric string. Associated with each user-id is a secret password. The correct password must be specified to gain access to the system. The password is not displayed when typed in response to the PASSWORD? prompt.

Command Qualifier: None.

Examples:

The user, Jones (with the password, Sam), requests access to the system. The system responds with an acknowledging message; the content of the message is determined by the system manager.

```
>LOGIN SAMPLE SESHUN

       TRAX VERSION 1.0A SYSTEM

GOOD MORNING
19-JUL-78 08:55 LOGGED ON TERMINAL TT4:
```

This login is equivalent to the first example but uses the prompts so that the password does not show on the terminal.

```
>LOGIN
USERID? SAMPLE
PASSWORD:

       TRAX VERSION 1.0A SYSTEM

GOOD MORNING
19-JUL-78 08:57 LOGGED ON TERMINAL TT4:
```

9.27 LOGOUT

The LOGOUT Command terminates user access to the system. The system aborts tasks and releases resources. The terminal then becomes available to other users.

Format:

LOGOUT

Prompt: None.

Command Parameter: None.

Command Qualifier: None.

Note:
When a nonprivileged user issues the LOGOUT command, the system aborts active tasks initiated by the user, dismounts the user's private volumes, and deallocates the user's private devices. The system then issues an acknowledging message whose content is determined by the system manager.

Example:

```
>LOGOUT
 TRAX
19-JUL-78 08:55 TT4:  LOGGED OFF
```

9.28 MACRO

The MACRO command assembles one or more MACRO source files into a single relocatable binary object module.

Format:

MACRO [/qualifiers] file-spec [file-qualifiers] +. . .

Command Qualifiers:	*Default:*
/LIST [: list-file-spec]	
/NOLIST	/NOLIST
/OBJECT [: object-file-spec]	
/NOOBJECT	/OBJECT
/[NO] CROSS_REFERENCE	/NOCROSS_REFERENCE
/SWITCHES (: switch-list)	

File Qualifiers

/PASS:n
/LIBRARY

Prompts:

FILE? file-spec [/file-qualifiers] + . . .

Command Parameter:

file-spec	Specifies a file that contains MACRO source code. Multiple input file-specifications must be concatenated with a plus (+) sign. Specifications must include a file name. If the file type is omitted. MAC is assumed, unless /LIBRARY is used, .MLB is assumed. No wildcards are allowed.

Command Qualifiers:

/NOLIST

Specifies that an assembly listing is not to be generated. This is the default.

/LIST [: list-file-spec]

Specifies that an assembly listing will be generated. If list-file-spec is given, then that file is not spooled; otherwise the listing is printed. The default file name is the name of the source file in the list, and the default file type is .LST.

/NOOBJECT

Specifies that an object module is not generated.

/OBJECT [: object-file-spec]

Specifies that an object module is to be generated. This is the default. The default name given to the object module file is taken from the last source file name in the list and is given the filetype .OBJ. This default name may be overridden by supplying the optional object-file-spec.

/[NO] CROSS_REFERENCE

/Specifies whether a cross reference listing is to be appended to the listing file. This implies use of the LIST qualifier. The default is /NOCROSS_REFERENCE.

/SWITCHES (: switch-list)

Enables you to pass to MACRO the standard listing options you wish to use.

Switch-list has the form:

switchl:argl . . . switch:argn

If switch is /LI or /NL, the arguments are:

SEQ	LOC	BIX	BEX	SRC	COM
MD	MC	ME	MEB	CND	LD
TOC	SYM	TTM			

If switch is /EN or /DS, the arguments are:

ABS	AMA	CDR	FPT	GBL
LC	LSB	PNC	REG	

They are defined in the TRAX MACRO Reference Manual.

File Qualifiers:

The file qualifier may be one or both of these.

/PASS:n	Specifies that the file is only to be assembled during the pass specified (n may be either 1 or 2).
/LIBRARY	Specifies that the file is a macro library file. /LIBRARY is not allowed on the last source file in the list. The default file type is MLB.

Note:
Library files must appear in a fixed order with respect to the source.

Example:

This command assembles the input files B, C, and D.MAC (using the necessary macros defined in A.MLB), creating the object module OBJMOD.OBJ and a listing file D.LST which will be spooled.

```
>MACRO/LIST/OBJECT:OBJMOD
FILE? A/LIBRARY+B+C+D
```

9.29 MERGE

The MERGE command merges records currently in one file with the records of another existing file; the receiving file must have relative or indexed organization.

Format:

```
MERGE [/LOG [: log-file-spec] input-file-spec [/qualifier] ]
        output-file-spec [/qualifier]
```

Command Qualifiers:	*Default:*
/LOG [log-file-spec]	See qualifier description.
Input-file Qualifier:	
/SEQUENTIAL /RELATIVE /INDEXED [/KEY:NUMBER:n]	/SEQUENTIAL
Output-file Qualifier:	
/INDEXED /RELATIVE	One is required.

Prompts:

```
FILE? Input-file [/qualifier]
INTO? output-file/qualifier
```

Command Parameters:

Input-file-spec	Specifies the file containing the records to be merged.
Output-file-spec	Specifies the file that receives the new records.

Command Qualifier:

/LOG [:log-file-spec]	The qualifier is optional; if specified, a log of all error messages is created during the merge sequence. An example is a listing of all records taken from the input-file that could not be merged, due to a duplicated key being detected when duplicate keys are not supported. If the file specification is omitted, the file name defaults to T10: and the log is printed in the OUTPUT stream.

Input-File Qualifier:

/SEQUENTIAL	Specifies a sequential file. This is the default.
/RELATIVE	Specifies a relative file.
/INDEXED	Specifies an indexed file. If specified, the /KEY qualifier is also required.
/KEY:NUMBER:n	The order of the record extraction may be specified by the use of the KEY qualifier. This is meaningful only when the /INDEXED qualifier is used.

Output-File Qualifier:

An output-file qualifier is required.

/INDEXED	Specifies an indexed structured file.
/RELATIVE	Specifies a relative structure file.

Note:
Wildcards are not permitted in either file specification parameter.

Example:

This example merges all records from the sequential file PROLL1.SEQ to the indexed file PROLL2.NDX.

```
>MERGE
FILE? PROLL1.SEQ
INTO? PROLL2.NDX/INDEXED
```

9.30 MESSAGE

The message command displays a message at a specified terminal. Two bells are sounded at the receiving terminal. The message is preceded by the date and the originating terminal number.

Format:

MESSAGE [/qualifier] [message]

Command Qualifiers:

/TERMINAL:TTn

Prompt:

TERMINAL? TTn
MESSAGE? message

Command Parameter:

message	The message is any combination of alphanumeric and control characters. Up to 68 ASCII characters can be specified. The string is terminated by the carriage return.

Command Qualifiers:

/TERMINAL:TTn:	Sends the message to terminal TTn. The terminal must be logged into the system for the message to be displayed. The system ignores the command when the terminal is not logged into the system. The /TERMINAL qualifier is required, although the terminal specification is promptable, that is, the TTn: value.

Examples:

1.
```
>MESSAGE/TERMINAL:TT4:  LOAD DISK1 ON DB3:
```

On the receiving terminal the message appears as:

```
19-JUL-78 09:04          FROM TT3:        TO TT4:
LOAD DISK1 ON DB3:
```

2.
```
>MESSAGE/TERMINAL
TERMINAL? TT4
MESSAGE? TIME TO GO HOME
```

At terminal 4, the message appears as:

```
19-JUL-78 09:05          FROM TT3:        TO TT4:
TIME TO GO HOME
```

9.31 MOUNT

Use the MOUNT command to logically connect a volume to the file system. The system also ensures the device is on-line. The system writes information on the volume permitting subsequent I/O access. Before each file access to the volume, I/O access is verified.

You are allowed to mount volumes on those devices currently allocated to you; that is, your private devices.

For efficient resource management, volumes should be dismounted when they are no longer needed. (Refer to the DISMOUNT command for its use.)

Format:

MOUNT [/qualifiers] device-name volume-label

Command Qualifiers:	*Default:*
/EXTENSION:blocks	Pack default
/PROTECTION:code	See Command Qualifier.
/OVERRIDE:option(s)	
/OWNER: [ggg, mmm]	See Command Qualifier.
/UNLOCKED	
/SHOW	
/WINDOW:m	m=0
/DENSITY:bpi	bpi=800

Prompts:

```
DEVICE? device-name
VOLUME LABEL? volume-label
```

Command Parameters:

device-name	The device name of the device to mounted. A list of legal physical device names is presented in Section 3.2.1. A logical device name that was previously assigned to a physical device name is permitted.
volume-label	The volume-label is a name associated with each volume. The volume-label is mandatory for magnetic tape and disk volumes unless the /OVERRIDE qualifier is specified. When the volume-label is specified, the system verifies the correct volume-label used.

The command is ignored when an incorrect volume-label is specified. The volume-label may be up to 6 alphanumeric characters for magnetic tape volumes and up to 12 alphanumeric character for disk.

Command Qualifiers:

/EXTENSION:n

Specifies the number of blocks a disk file shall be extended if it exhausts its current space allotment. The n value is decimal.

This qualifier overrides the EXTENSION:n specified in the INITIALIZE command. If not specified, the INITIALIZE/EXT condition is the default.

/PROTECTION:code

Change the file protection access.

When either /PROTECTION is not specified or specific classes within the code are not specified, the default values are taken from the volume. See the INITIALIZE command for the volume default values. Also see Section 3.3.2 for a detailed description of how to form the code parameter.

/OVERRIDE:option(s)

The /OVERRIDE qualifier permits several MOUNT options to be ignored. The options are separated by comma and enclosed in parentheses if more than one is used.

The options and their functions are:

IDENTIFICATION Do not verify the volume identification. Note that when IDENTIFICATION is specified, no other /OVERRIDE option can be given.

LABEL Do not verify the magnetic tape volume label.

EXPIRATION DATE Override the expiration date on the magnetic tape volume.

/OWNER: [ggg, mmm]

Change the User File Directory (UFD) of the owner of the volume.

The group number, ggg, and the member number, mmm, are octal values from 0 to 377. The square brackets [ ] are required syntax.

/UNLOCKED

Permits read/write access to files on the volume. When the qualifier is not specified, no write access is permitted.

/SHOW

Displays the volume information at the issuing terminal.

/WINDOW:m	Overrides the number of mapping pointers allocated for disk file windows set up at the volume initialization. The range of m is from 0 to 9. This applies only to disk volumes.
/DENSITY:bpi	Set the magnetic tape density to either 1600 or 800 bits per inch (bpi).

Examples:

This example mounts the disk, DB0:, and verifies that the volume label is VOLNAME.

```
>MOUNT DB0: VOLNAME
```

This example mounts disk, DB2: and verifies the volume label is SYS001. Specifies default file protection for this volume as [RW, RWED, RWED, R]. Permits the system read and write access and permits the group read, write, extend, and delete access. The owner and world access are unchanged.

```
>MOUNT/PROTECTION:(SYSTEM:RW,GROUP:RWED)
DEVICE? DB2:
VOLUME ID? SYS001
```

9.32 $ON

The $ON command specifies an action to be taken if a subsequent batch command returns an error status equal to or greater than a specified level.

Format:

$ON status-level THEN action

Prompt: None.

Command Parameters:

Status-level is one of the following:

WARNING
ERROR
SEVERE_ERROR

Action is one of the following:

CONTINUE

GOTO label	(Label is an alphanumeric string and must appear, together with a colon, in front of a subsequent command.)

STOP

Command Qualifiers: None.

Notes:

1. $ON ERROR STOP is assumed by default at the beginning of a batch job.

2. The THEN action is taken if a subsequent command returns a status level equal to or greater than the status level specified in the $ON command.

3. An $ON command remains in force until superseded by another $ON command, until an ON condition is met, or until end-of-job, whichever occurs first.

4. A $ON command can be suspended by a $SET NOON command and can be later reinstated by a $SET ON command.

Example:

The $ON and $MACRO commands are executed. If the assembly is completed with nothing worse than a warning, the job proceeds to $LINK. If the linking is completed with nothing worse than a warning, the job proceeds to $RUN. If any of these commands produces a status-level of ERROR or SEVERE_ERROR, the job is stopped; in this case, all remaining commands in the file are skipped.

```
$JOB
$ON ERROR STOP
$MACRO MYPROG
$LINK MYPROG
$RUN MYPROG
$EOJ
```

9.33 PRINT

The PRINT command causes one or more files to be printed on the line printer. It defines a printing job to be placed in the print queue.

Format:

PRINT [/qualifiers] file-spec [/qualifiers], . . .

Command Qualifiers:	*Default:*
/[NO] DELETE	/NODELETE
/COPIES:n	/COPIES:1
/QUEUE:queue-name	/QUEUE:PRINT
/UPPERCASE	/UPPERCASE
/LOWERCASE	
/[NO] ORIGINAL	/NOORIGINAL
/[NO] WIDE	/NOWIDE
/PAGES:n	Pages unlimited

/JOB: jobname	JOB: First six characters of first filespec
/PRIORITY:n	/PRI:50
/FORMS:n	/FORMS:0
/LENGTH:n	No implied form feeds
/[NO] RESTART	
/[NO] FLAG_PAGE	/NOFLAG_PAGE
/AFTER: (dd-mmm-yy hh:mm)	Present time

File Qualifiers:

/[NO] DELETE
/COPIES:n
/[NO] ORIGINAL

Prompt:

FILE? file-spec [/qualifier], . . .

Command Parameter:

file-spec	Specifies a file to be printed. If no file type is included in the file specification, the default file type is .LST.

Command Qualifiers:

/[NO] DELETE	Instructs the system to delete all files after printing. The default is /NODELETE.
/COPIES:n	Specifies the number of file copies to be printed. The value of n is an integer from 1 to 32 (decimal), with a default of 1.
/UPPERCASE	Specifies that an uppercase-only printer is sufficient for printing the job.
/LOWERCASE	Specifies that a printer capable of printing both upper and lower case characters is needed for this job.
/QUEUE:queue-name	Specifies the name of the queue in which the job is to be placed. If this qualifier is omitted, the job is placed in the queue named PRINT.
/[NO] WIDE	Specifies whether a wide printer is required. A wide printer has at least 132 characters per line.

/PAGES:n	Specifies the maximum number of pages that the job may produce. Default is unlimited.
/JOB:jobname	Specifies the name of the job to be placed in the queue. If omitted, the system will assign a job name based on the first six characters of the first file name.
/PRIORITY:n	Specifies the queue priority level of the print job. The argument n must be an integer in the range 1 to 250, and 250 is the highest priority.
/FORMS:n	Specifies the forms attribute of a print job. The FORM option complements the LENGTH option in defining the basic vertical boundaries and margins of an individual form. The FORMS option indicates, directly or indirectly, the size of the form. Usually, this is the number of print lines between perforations. The default forms attribute is n=0, which indicates that form-feed processing will be handled by the printer. Values of n from 1 to 255 indicate that the software will handle form-feed processing. Values of n from 1 to 66 denote the actual number of lines by default, although they can be redefined by the installation. Values of n greater than 66 require installation definition.
/LENGTH:n	Specifies the number of lines that can be printed on a form page. If, while processing the print job, form-feed characters are not found in the file within n lines of the last form feed, a form feed is generated. Thus, if the FORMS option indicates a form size of 66 lines, and LENGTH specifies that no more than 60 lines may be printed per form page, the combination of options implies a bottom margin of six lines. By default, the system generates no form feeds; this is equivalent to specifying /LENGTH:0.
/[NO] ORIGINAL	Indicates whether or not to make temporary copies of files that exist on private volumes. /ORIGINAL requests that no copy be made.
/[NO] RESTART	Specifies whether a job can be restarted from the beginning following an interrupt, such as running out of paper.

/[NO] FLAG_PAGE	/NOFLAG_PAGE suppresses the flag page before each file in the job. /NOFLAG_PAGE is the default.
/AFTER: (dd-mmm-yy hh:mm)	Specifies the date and the time after which the job will become eligible for dispatching to some print processor.

File Qualifiers:

/[NO] DELETE	Specifies whether or not the file is to be deleted after printing.
/COPIES:n	Specifies the number of list copies to be produced for the file.
/[NO] ORIGINAL	Indicates whether or not to use a temporary copy of the file. /ORIGINAL directs that no copy be made.

Notes:

1. If /[NO] DELETE, /COPIES:n, or /[NO] ORIGINAL is specified as both a command qualifier and a file qualifier, the file qualifier overrides the command qualifier for that file. Any of these qualifiers given at command level sets a default qualifier that applies to all files unless overridden by a file qualifier.

2. The /QUEUE qualifier is optional. If it is omitted, the job is added to the default print queue, named PRINT.

Example:

The following command prints two copies each of every file in the directory having the file name RESULT. It also prints four copies of the file PRIME.DAT. The job is to be queued on the queue NIGHT.

```
>PRINT/COPIES:2/QUE:NIGHT
FILE? RESULT.*,PRIME.DAT/COPIES:4
```

9.34 PURGE

The PURGE command removes older versions of one or more specified files, retaining one or more latest versions. See also the description of the DELETE command earlier in this chapter.

Format:

PURGE [/qualifier] file-spec [, . . .]

Command Qualifier:	*Default:*
/KEEP:n	/KEEP:1

Prompt:

FILE? file-spec [, . . .]

Command Parameter:

file-spec	The file specification of the file whose early versions are to be deleted from the Directory. Each file specification must include a file name and a file type, but not a version number.

Command Qualifier:

/KEEP:n	Specifies that the n latest versions of the file shall be retained. The system locates the highest version number associated with the file specifications and deletes all versions of the file with a version number lower than the highest version number minus n. The default value of 1 is assigned when n is either not specfied or 0. For example, assume you specify n as 3 and the system determines that the latest version number of the file is 7. This requests that versions of the file version numbers of 4 or less be deleted. The system keeps version 7 and also keeps versions 6 and 5, if they have not been removed by an earlier PURGE or DELETE operation.

Notes:

1. If the /KEEP qualifier is not specified, it is equivalent to specifying /KEEP:1.

2. Wildcards are allowed in the directory, file name, and file type components of the file specification.

3. You may purge only those files for which you have delete access rights.

Examples:

This example purges all versions of TEST.TMP except the highest numbered (latest) version.

```
>PURGE TEST.TMP
```

This example purges all but the latest two versions of SRTRT.B2S.

```
>PURGE/KEEP:2 SRTRT.B2S
```

9.35 RENAME

The RENAME command renames an existing file.

Format:

RENAME old-file-spec new-file-spec

Prompts:

OLD? old-file-spec
NEW? new-file-spec

Command Parameters:

old-file-spec	Specifies an existing file. The file specification must include a file name and a file type.
new-file-spec	Specifies the new name for the existing file. The file specification must include a file name and a file type.

Command Qualifiers: None.

Notes:

1. Both specifications must have the same device (since files may not be renamed across devices).

2. Wildcards are allowed in the file type and file version fields of each file specification. Wildcards appearing in one file specification must appear in the corresponding fields of the other file specification.

3. If a version number is omitted from new-file-spec, the version number of the old-file-spec is used by default.

4. You can RENAME a file into another UFD, protection conditions permitting. As a general rule, if you are authorized to create a file in a UFD, you are also authorized to rename files stored under that UFD. Also, you can use RENAME to change the UFD of a file.

Examples:

This example renames the fourth version of the OLD.TMP to NEW.TMP.

```
>RENAME
OLD? OLD.TMP;4
NEW? NEW.TMP;1
```

9.36 RUN

The RUN command permits the system to install a task, run it, and—upon completion—remove the task from the system. Use the ABORT command to terminate an active task.

Format:

RUN[/TASK:task-name] file-spec

Command Qualifier:	*Default:*
/TASK:task-name	See Qualifier description.

Prompt:

FILE? file-spec

Command Parameters:

file-spec	Specifies a file specification referring to a linked program, or task. When the file name is preceded by the dollar sign ($), the system searches the System File Directory for the file. When the file name is not preceded by either the dollar sign ($) symbol or a UFD, the system searches the default User File Directory for the file. The default file type is .TSK.

Command Qualifiers:

/TASK:task-name	Specifies the task name to assign when installing the task. The name may be up to 6 characters. The default name is TTnn where nn is the unit number of the requesting terminal.

Examples:

This example immediately loads and executes the latest version of the file TSK1.TSK from the user's current default device and directory. The system removes the task when either the task runs to completion or the user aborts the task.

```
>RUN TSK1
```

This example installs and executes the task file DEMO.TSK from the default UFD and names the task, TEST. The system names tasks initiated through the RUN command, by default, to be TTnn where nn is the user's terminal number.

```
>RUN/TASK:TEST DEMO.TSK
```

9.37 SET

The SET command enables you to alter dynamically certain characteristics of your terminal, files, devices, queued job, and operating environment.

Format:

```
>SET function
```

Prompt:

```
FUNCTION? function
```

Command Parameters:

function	Specifies one of the following: DEFAULT DEVICE [NO]ON PROTECTION QUEUE TERMINAL The use of each of these options is described in ensuing sections.

Command Qualifiers: None.

Notes:

1. The function parameter further defines the SET command. Each SET function has its own command syntax, as described in the following subsections.

2. The SHOW command complements the SET command. It enables you to ascertain the current characteristics before you modify them.

9.37.1 SET DEFAULT

The SET DEFAULT command establishes the user default system device or the User File Directory, or both, at the issuing terminals. These defaults are used when referencing files within command lines, overriding previous defaults.

Format:

>SET DEFAULT [device-name] [ufd]

Prompts:

FUNCTION? DEFAULT
DEFAULT DEVICE NAME AND/OR DIRECTORY? [device-name] [ufd]

Command Parameters:

device-name	Specifies the device name of the device you wish to use as the default device in subsequent commands.
ufd	Specifies the new default User File Directory in the format: [ggg,mmm] The group number, ggg, and the member number, mmm, are octal values from 0 to 377. The brackets and comma are required. If the device-name parameter is present, the UFD must immediately follow the colon that marks the end of the device-name parameter.

Command Qualifiers: None.

Notes:

1. Changing the default device and UFD does not change your UIC; your UIC is determined when you log in.
2. If you change the default UFD, you will be accessing files under different protection access codes. Under normal default protection values, you will have read access to all files regardless of group number, and full access to files stored under the group number in your UIC. See the following example for further clarification, and also see the description of the SET PROTECTION command.

Example:

The following directory listing indicates that several files with file name DEMO are stored in the UFD [40,40]. This UFD is the current default, determined from the UIC established at log in time.

```
>DIRECTORY DEMO.*

DIRECTORY DB0:[40,40]
19-JUL-78 10:26

DEMO.B2S;1          1.          18-JUL-78 16:53
DEMO.OBJ;1          2.          18-JUL-78 16:53
DEMO.CMD;1          1.          18-JUL-78 16:53
DEMO.ODL;1          1.          18-JUL-78 16:53
DEMO.TSK;1          15.      C  18-JUL-78 16:53
DEMO.MAP;1          12.         18-JUL-78 16:53

TOTAL OF 32./32. BLOCKS IN 6. FILES
```

When you change the default UFD to include a different group number, you can read, but cannot write, files in that UFD. (The UIC is still [40,40].)

```
>SET DEFAULT DB0:[350,230]
>COPY [40,40]DEMO.*   *.*
COP -- OPEN FAILURE ON OUTPUT FILE
DB0:[350,230]DEMO.B2S -- HANDLER ERROR CODE -16.
```

However, when you set the default UFD to a value with the same group number as your UIC, you can write files in that directory under normal protection conditions. The final DIRECTORY listing shows that the copy operation was successful.

```
>SET DEFAULT DB0:[40,41]
>COPY [40,40]DEMO.*  *.*
>DIRECTORY DEMO.*

DIRECTORY DB0:[40,41]
19-JUL-78 10:33

DEMO.B2S;2          1.          19-JUL-78 10:33
DEMO.OBJ;2          2.          19-JUL-78 10:33
DEMO.CMD;2          1.          19-JUL-78 10:33
DEMO.ODL;2          1.          19-JUL-78 10:33
DEMO.TSK;2          15.      C  19-JUL-78 10:33
DEMO.MAP;2          12.         19-JUL-78 10:33

TOTAL OF 32./32. BLOCKS IN 6. FILES
```

9.37.2 SET DEVICE

The SET DEVICE command dynamically alters the attributes of a specified device.

Format:

SET DEVICE:device-name option

Prompts:

FUNCTION? DEVICE
DEVICE? device-name
ATTRIBUTE? option

Command Parameters:

device-name	Specifies the device whose characteristics are to be changed.	
option	Specifies one of the following:	
	WIDTH:n	Applies to terminals and printers, specifying the character width of the output medium. The value of n can be octal or decimal. Octal values are expressed as an integer followed by a space, and decimal values are expressed as an integer immediately followed by a decimal point. For example 72 is octal and 72. is decimal.
	[NO]WRITECHECK	Specifying WRITECHECK requests verification of write operations by means of an automatic read after write. NOWRITECHECK is the default.

Examples:

1. This sets the line buffer size of the LPO: to 132 characters.

```
>SET DEVICE:LPO:  WIDTH:132
```

2. In this example, each record written on the device DB0: will be read back and verified.

```
>SET DEVICE:DB0: WRITECHECK
```

9.37.3 $SET[NO]ON

The $SET NOON and $SET ON commands are used only in batch jobs. The $SET NOON command suspends the influence of an $ON command until a $SET ON command reinstates it.

Format:

$SET [NO]ON

Notes:

1. The command has no parameters or qualifiers.
2. $SET NOON is meaningful only if a $ON command exists earlier in the file.
3. $SET ON is meaningful only if a $SET NOON command has been executed.

9.37.4 SET PROTECTION

The SET PROTECTION command alters the protection access rights for a specified set of files.

Format:

SET PROTECTION [(] file-spec [, . . .)] code

Prompts:

FUNCTION? protection
FILE? [(] file-spec[, . . .)]
PROTECTION? code

Command Parameters:

file-spec	Specifies the file specifications of one or more files whose access protection rights are to be altered. If more than one file-spec is given, the set of file-specs must be enclosed within parentheses and separated by commas. The files must already exist. Each file-spec must include a file name and a file type. If a file version number is omitted, the highest version is used.
code	Specifies the new access protection rights of the files. See Section 3.3.2 for information on file protection codes.

Command Qualifiers: None.

Notes:

1. The code parameter must be formed as for the /PROTECTION qualifier of the CREATE command.
2. Only one code parameter is permitted in any SET PROTECTION command. All files specified have their own protection set according to the code parameters.

Example:

This alters the system access rights of the file A.TEMP to Read and Write and group access rights to Read, Write, Extend, and Delete.

```
>SET PROTECTION A.TMP   (SYSTEM:RW,GROUP:RWED)
```

This changes the protection of A.TMP and B.TMP as described in the previous example.

```
>SET PROTECTION
FILE? (A.TMP,B.TMP)
PROTECTION? (SY:RW,GR:RWED)
```

9.37.5 SET QUEUE

The SET QUEUE command modifies one or more attributes of a job in a print or batch queue (that is, attributes assigned to a job by the PRINT or SUBMIT command). You are allowed to modify only those jobs that you have placed in the queue.

The command is also used to hold a job in a queue and to release it subsequently for normal processing.

Format:

SET QUEUE queue-id option [, . . .]

Prompts:

FUNCTION; QUEUE
QUEUE NAME OR ENTRY? queue-id
OPTIONS? option(s)

Command Parameters:

queue-id — Identifies the queue to be affected. Two forms are allowed:

- queue-name — Specifies a print queue or batch queue. The command SHOW QUEUE ALL displays the names of queues recognized by the system.
- ENTRY: (n,n) — Specifies the job entry number, as internal identification assigned by the system when you originated the job. If you specify this, you need not specify queue-name or job-name elsewhere in the statement. The SHOW QUEUE command with the FULL option displays job entry identifiers. Both components of the entry number are octal. The word

ENTRY cannot be appreviated. It is a standard system queue name, not a DCL keyword.

option

Specifies one or more of the following:

JOB:job-name	Specifies the name of the job that you wish to modify. If omitted, the job name is assigned by the system, based on your user identification. When ENTRY is specified, the JOB option is not permitted.
UPPERCASE	Specifies that the print job requires only a printer with an uppercase character set.
LOWERCASE	Specifies that the print job requires a printer with upper - and lower/case characters.
[NO]WIDE	WIDE specifies that a wide line printer (132 characters) is required. NOWIDE negates an outstanding WIDE specification.
PAGES:n	Specifies that the print job should be aborted if it exceeds n pages.
PRIORITY:n	Specifies the queue priority level of the job.
FORMS:n	Specifies the forms attribute of a print job. The FORM option complements the LENGTH option in defining the basic vertical boundaries and margins of an individual form. The FORMS option indicates, directly or indirectly, the size of the form. Usually, this is the number of print lines between perforations. The default forms attribute is n=0, which indicates that form-feed processing will

	be handled by the printer. Values of n from 1 to 255 indicate that the software will handle from-feed processing. Values of n from 1 to 66 denote the actual number of lines by default, although they can be redefined by the installation. Values of n greater than 66 require installation definition.
LENGTH:n	Specifies the number of lines that can be printed on a form page. If, while processing the print job, form-feed characters are not found in the file within n lines of the last form feed, a form feed is generated. Thus, if the FORMS option indicates a form size of 66 lines, and LENGTH specifies that no more than 60 lines may be printed per form page, the combination of options implies a bottom margin of six lines. Default produces no implied or generated form feeds, and is equivalent to specifying LENGTH:0.
[NO]RESTART	Specifies whether the job can restart from the beginning if stopped.
[NO]FLAG_PAGE	Enables or disables the printing of flag pages before files in the print job.
AFTER:(DD-MM-YY HH:MM)	Specifies the date and time after which the job may be processed.
HOLD	Unconditionally holds the job in the queue, delaying normal processing until a RELEASE is given even though other conditions could indicate that it should be processed.

RELEASE	This places the job in the waiting job list. The job is processed when it reaches the top of the waiting job list.

Command Qualifier: None.

Examples:

This queued batch job TEST on directory [40, 41] is given a modified priority of 120. BATCH is the queue-name.

```
>SET QUEUE BATCH JOB:[40,41]TEST,PRIORITY:120
```

The queued entry (112, 223) is changed to accommodate forms of type 1, as defined by the installation.

```
>SET QUEUE ENTRY:(112,223) FORMS:1
```

9.37.6 SET TERMINAL

The SET TERMINAL command establishes or changes the attributes of your terminal.

Format:

>SET TERMINAL [terminal-name] option

Prompt:

FUNCTION? TERMINAL
ATTRIBUTE? option

Command Parameters:

terminal-name	Optional; if entered, it must be TI: (your terminal).	
option	One of the following:	
	TYPE:value	Specify the type of terminal. If the terminal is a video display, specify SCOPE; if a printed display, specify NOSCOPE.

LOWERCASE	Recognize lowercase and uppercase characters on input.
UPPERCASE	Recognize uppercase characters. Lowercase characters are converted to uppercase on input.
[NO] SLAVE	SLAVE establishes the terminal as a device that cannot enter unsolicited input. Only information requested from a task is recognized. NOSLAVE removes this restriction.
[NO]HOLD_SCREEN	Enable/disable hold screen mode at the specified terminal. In hold screen mode the terminal displays a full screen of data each time the scroll key is pressed.
SPEED:(n, m)	Establishes the receive baud rate (n) and the transmit baud rate (m) of the terminal. See your system manager for the possible baud rates on your terminal.
[NO] ESCAPE SEQUENCE	Enable/disable the recognition of escape sequences from the specified terminal.

Command Qualifiers: None.

Example:

This example sets the terminal to a printed output type terminal.

```
>SET
FUNCTION? TERMINAL
ATTRIBUTE? TYPE:NOSCOPE
>
```

9.38 SHOW

The SHOW command complements other commands, such as SET, by displaying at your terminal all information pertaining to your task, terminal, and devices that you may establish or alter.

Format:

SHOW function

Prompts:

FUNCTION? function

Command Parameters:

Function specifies one of the following:

ASSIGNMENTS [:option]
DAYTIME
DEFAULT
DEVICES
QUEUE
TASKS
TERMINAL

The use of each of these options is described in ensuing sections.

Command Qualifiers: None.

9.38.1 SHOW ASSIGNMENTS

The SHOW ASSIGNMENTS command displays local assignments currently in force at your terminal.

Format:

SHOW ASSIGNMENTS [:LOCAL]

Prompts:

FUNCTION? ASSIGNMENTS

Command Parameters: None.

Command Qualifiers: None.

Notes:

1. The string :LOCAL may be omitted without altering the effect of the command.

2. SHOW ASSIGNMENTS complements the ASSIGN command, displaying logical name assignments.

Example:

The following example shows the device DB1: to have a local assignment of MY0 = at terminal 4. The default log in assignment is also shown.

```
>SHOW ASSIGNMENTS
MY0:     DB1:     LOCAL   TI - TT4:
SY0:     SY0:     LOGIN   TI - TT4:
>
```

9.38.2 SHOW TIME

The SHOW TIME displays the time of day and the date.

Format:

SHOW TIME

Prompt:

FUNCTION? TIME

Command Parameters: None.

Command Qualifiers: None.

Example:

```
>SHOW TIME
13:17:05 19-JUL-78
>
```

9.38.3 SHOW DEFAULT

The SHOW DEFAULTS command displays your current default device name and directory.

Format:

SHOW DEFAULT

Prompt:

FUNCTION? DEFAULT

Command Parameters: None.

Command Qualifiers: None.

Example:

In this example, the response indicates that DB0 = is the current default device, and [350, 230] is the current directory.

```
>SHOW DEFAULT
   DB0:[350,230]
>
```

9.38.4 SHOW DEVICES

The SHOW DEVICES command displays the symbolic names and status of the devices. The display is made on the entering terminal.

Format:

SHOW DEVICES [option]

Command Qualifier: Default:

None

Prompt:

FUNCTION? DEVICES

Command Parameter:

option	The option can be any of the following words. The system displays all the options when no option is specified.
[NO] PUBLIC	Display only those devices allocated as either PUBLIC or NOPUBLIC.
TYPE device-name:	Display only the devices specified by the device name, such as DB:.
[NO] WRITECHECK	Display only those devices in either the WRITECHECK or NOWRITECHECK mode.
WIDTH: device-name	Display the buffer size of the specified device.

Command Qualifier: None

Examples:

This example displays the buffer size of LP0: as 132 decimal characters.

```
>SHOW DEVICE WIDTH:LP0:
BUF=LP0:00132.
>
```

This example describes the status of every device on the system.

```
>SHOW DEVICES
DB0:     PUBLIC   MOUNTED LOADED
DB1:     LOADED
DB2:     LOADED
DB3:     PUBLIC   MOUNTED LOADED
DR0:     LOADED
DR1:     MOUNTED LOADED
MM0:     LOADED
MM1:     LOADED
LP0:     DB0:     SPOOLED LOADED
TT0:     [1,1]    - LOGGED ON      LOADED
TT1:     LOADED
TT2:     LOADED
TT3:     LOADED
TT4:     [40,40] - LOGGED ON      LOADED
TT5:     [1,1]    - LOGGED ON      LOADED
TT6:     LOADED
TT7:     [350,227]        - LOGGED ON      LOADED
TT10:    [1,1]    - LOGGED ON      LOADED
NL0:
VT0:     LOADED
VT1:     LOADED
VT2:     LOADED
TI0:
CO0:     TT0:
CL0:     LP0:
SP0:     DB0:
LB0:     DB0:
SY0:     DB0:
```

9.38.5 SHOW QUEUE

The SHOW QUEUE command complements the SET QUEUE command. It enables you to display current information about your print and batch queues entries.

Format:

SHOW QUEUE [queue-id] [option [form-option]]

Prompts:

FUNCTION? QUEUE

Command Parameters:

queue-id

Identifies the queue or queues to be displayed. Three forms are allowed:

queue-name — Specifies the name of the queue, as defined in the SUBMIT and PRINT command.

ENTRY: (m, n) — Specifies the job entry number, an internal identifier assigned to each queue entry when the job is originated. If you specify this, you need not specify queue-name or job name elsewhere in the command. The word ENTRY cannot be abbreviated; it is a standard system queue name, not a DCL keyword.

ALL — Indicates that all existing queues of a given type are to be displayed. ALL may be followed with the option PRINT or BATCH, indicating that only print or batch queues should be displayed. The word ALL cannot be abbreviated; it is a standard system queue name, not a DCL keyword.

option

Further defines the SHOW QUEUE command. Possible options are:

JOB: [uic] jobname — Requests information about a particular job originated by a specified user.

USER: [uic] — Requests information on all queue entries for a specified user.

	NUMBER	Requests the number of entries in the specified queue or queues.
	ALL	Requests display of all entries in the specified queue or queues.
	PRIORITY:n	Shows all job entries at the specified priority level.
	FORMS:n	Shows all job entries with the specified forms type.
	BATCH	If used in conjunction with the queue-name ALL, only BATCH queues are listed.
	PRINT	If used in conjunction with the queue-name ALL, only PRINT queues are listed.
form-option	Specifies the form of the information to be listed.	
	BRIEF	Specifies a limited listing, consisting only of the names of the entries. In many cases, this only confirms the presence of the entries specified by the command.
	FULL	Specifies a complete report on all queues and entries specified or implied by the command.

Command Qualifiers: None.

Examples:

There are many combinations of parameters for this command. The following sequence of examples illustrates the type of output you can expect for the various forms of SHOW QUEUE:

```
>SHOW QUEUE ALL
  PRINT QUEUES
    PRINT
    LPQ0
    TEST
    BAP0
  BATCH QUEUES
    BATCH
    TRXKIT  STOPPED
    SURVEY
    CHRIS
>SHOW QUEUE BATCH ALL BRIEF
  BATCH QUEUES
    BATCH
>SHOW QUEUE BATCH ALL FULL
  BATCH QUEUES
    BATCH
      ASSIGNED PROCESSORS
        BAP0
      HELD JOBS
        1  [1,1]SAMPLE  ENTRY:(12220,20663)  TI:TT37:  PRI:50  REST:Y
          PRINT:Y
          FILES
            DB0:[1,1]SAMPLEFDF.BLD;2  ENTRY:(12300,120664)  DEL:N
        2  [350,227]TRAXAP  ENTRY:(2140,20011)  TI:TT7:  PRI:50  REST:Y
          PRINT:Y
          FILES
            DB0:[350,227]TRAXAPG.CMD;2  ENTRY:(2160,120012)  DEL:N
        3  [40,40]BTST    ENTRY:(4640,10604)  TI:TT4:  PRI:50  REST:Y
          PRINT:Y
          FILES
            DB0:[40,40]BTST.CMD;1  ENTRY:(4660,110605)  DEL:N
        4  [1,1]CONT    ENTRY:(6040,7461)  TI:TT1:  PRI:50  REST:Y
          PRINT:N
          FILES
            DB0:[1,1]CONT.CMD;2  ENTRY:(6060,107462)  DEL:N
        5  [1,1]CONT    ENTRY:(5020,7455)  TI:TT1:  PRI:50  REST:Y
          PRINT:N
          FILES
            DB0:[1,1]CONT.CMD;1  ENTRY:(5540,107456)  DEL:N
        6  [300,307]BTCH01  ENTRY:(4500,7167)  TI:TT31:  PRI:50  REST:Y
          PRINT:Y
          FILES
            DB0:[300,307]BTCH01.BIS;1  ENTRY:(4520,107170)  DEL:N
        7  [200,200]DIALOG  ENTRY:(5560,6457)  TI:TT1:  PRI:50  REST:Y
          PRINT:N
          FILES
            DB0:[200,200]DIALOG.CMD;3  ENTRY:(5600,106460)  DEL:N
        8  [200,200]DIALOG  ENTRY:(3100,4261)  TI:TT37:  PRI:50  REST:Y
          PRINT:N
          FILES
            DB0:[200,200]DIALOG.CMD;2  ENTRY:(3120,104262)  DEL:N
```

```
 9  [1,1]TEST     ENTRY:(2060,1002)  TI:TT37:  PRI:50  REST:Y
   PRINT:N
   FILES
     DB0:[1,1]TEST.CMD;1  ENTRY:(2100,101003)  DEL:N
     DB0:[1,1]TEST.CMD;1  ENTRY:(2120,101004)  DEL:N
10  [1,1]TEST     ENTRY:(2000,465)  TI:TT1:  PRI:50  REST:Y
   PRINT:N
   FILES
     DB0:[1,1]TEST.CMD;1  ENTRY:(2020,100466)  DEL:N
     DB0:[1,1]TEST.CMD;1  ENTRY:(2040,100467)  DEL:N
>
```

9.38.6 SHOW TASKS

The SHOW TASKS command displays information about installed tasks. The information displayed is dependent upon the option selected.

Format:

SHOW TASKS status [display] [task-name]

Prompts:

FUNCTION? TASKS
ACTIVE OR INSTALLED? status

Command Parameters:

status

Specify either ACTIVE OR INSTALLED. When INSTALLED is specified, both active and dormant tasks are displayed.

display

Either FULL, ALL, or BRIEF can be specified. BRIEF displays, on your terminal, a list of the task-names. FULL displays, on your terminal, active task-names and their status.

When not specifying "display", BRIEF is used.

The FULL display contains the following information for each task:

- Task name
- Task control block physical address (octal)
- Partition name;
- Partition control block physical address (octal)
- Partition base and limit physical addresses (octal)
- Task's running priority and default priority;
- Task status flags;
- TI terminal physical device name;

- I/O count (decimal);
- Task local event flags, and
- Task registers and Processor Status Word (memory resident tasks only).

Flags prefixed by a minus (-) sign indicate the complementary status. That is, -CHK indicates that the task is not checkpointable.

When a task is not in memory (the OUT flag is displayed), the contents of the PC, PS, and the registers are not displayed.

STATUS	DESCRIPTION
ABO	Task is being aborted.
ACP	Task is an ancillary control processor.
AST	Task is processing an AST.
BFX	Task is being fixed in memory.
CAF	Checkpoint space allocation failure occurred.
CAL	Checkpoint space is allocated in task image.
CHK	Task is checkpointable.
CKD	Task checkpointable is disabled.
CKP	Task is checkpointed.
CKR	Task checkpoint request pending.
DST	Task ASTs are disabled.
EXE	Task is in execution.
FXD	Task is fixed in memory.
HLT	Task is being terminated.
MCR	Task was activated by MCR.
MSG	Task was aborted and waiting for TKTN message.
NRP	Task is mapped to nonresident partition.
NSD	Task cannot receive data (no send data allowed).
PMD	Suppress task post mortem dump abort.
OUT	Task is out of memory.
PRV	Task is privileged.
RDN	Task I/O is being run down.
REM	Task is to be removed on exit.
ROV	Task has resident overlays.

SLV	Task is slave.
SPN	Task is being suspended.
SPNA	Task was suspended prior to AST.
STP	Task stopped for terminal input.
STPA	Task stopped prior to AST.
TIO	Task is waiting for terminal input.
WFR	Task is in a "wait-for" state.
WFRA	Task was in a "wait-for" state before AST.

task-name — When the task-name is specified, only that task information is displayed. When the task-name is omitted, all tasks are displayed.

Command Qualifiers: None.

Examples:

This example lists the tasks active at your terminal.

```
>SHOW TASKS ACTIVE BRIEF
SHOT4
>
```

This example lists all tasks currently active in the system.

```
>SHOW TASKS ACTIVE ALL
...LDR
SHOT4
F11ACP
DB3ACP
QMG...
LPP0
BAP0
EDIT7
EDIT14
EDIT34
TT6
TNXT6
TT5
EDIT27
EDIT10
>
```

9.38.7 SHOW TERMINAL

Complements the SET TERMINAL command; displays a list of terminals for which the specified attribute is an established feature, or displays an attribute of a specified terminal.

Format:

SHOW TERMINAL option

Prompts:

FUNCTION? TERMINAL
ATTRIBUTES? option

Command Parameter:

option — Specifies one of the following items:

TYPE:term	Specify the type of terminal, where "term" can be SCOPE or NOSCOPE.
LOWERCASE	Display those terminals that recognize lowercase and uppercase characters.
UPPERCASE	Display those terminals that recognize uppercase characters only.
[NO] PRIVILEGED	Display the terminals in privilege mode.
[NO] SLAVE	Display the terminals in slave mode.
[NO] REMOTE	Display those terminals in REMOTE mode.
[NO] HOLD_SCREEN	Display the terminals in hold screen mode. When in hold screen mode, the terminal displays a full screen of data each time the scroll key is pressed. This is useful when displaying a file on a scope terminal.
[NO] ESCAPE_SEQUENCE	Display those terminals in the escape sequence mode.

SPEED:Tnn	Display the specified terminals' receive and transmit baud rates in the "transmit: receive" format. The terms "transmit" and "receive" are in reference to the terminal's ability to transmit and receive. When Tnn is not specified, the speed of the issuing terminal (TI) is shown.

Command Qualifier: None.

This example displays the transmit and receive rates of terminal 4.

```
>SHOW TERMINAL SPEED:TT4
SPEED=TT4:300:300
>
```

This example lists each terminal with a printer rather than a CRT display medium.

```
>SHOW TERMINAL TYPE:NOSCOPE
NOCRT=TT0:
NOCRT=TT4:
NOCRT=TT6:
NOCRT=VT0:
NOCRT=VT1:
NOCRT=VT2:
>
```

9.39 SORT

The SORT command invokes the SORT utility of TRAX. This utility sorts the contents of an input file into a sequence indicated by the SORT command, and writes the sorted contents into an output file.

Two types of SORT are allowed, a record sort or a tag sort.

A record sort produces a reordered file by examining the specified control keys and directly copying entire records to the output file as required.

A tag sort produces a reordered file by extracting the control keys into the proper order. Then the record pointer associated with each key is used to reaccess the input file randomly to produce the sorted output file. The tag sort is possible only when the input file resides on a disk.

See the TRAX SORT Reference Manual for further details.

Format:

SORT [/qualifiers] input-file-spec [/file qualifiers]

Command Qualifiers	*Default*
/ALLOCATION:n	See qualifier
/BLOCKSIZE:n	/BLOCKSIZE:512
/BUCKETSIZE:n	/BUCKETSIZE:1
/[NO] CONTIGUOUS	/NOCONTIGUOUS
/DEVICE:device-name	See qualifier
/FILES:n	/FILES:5
/FORMAT:type:n	See qualifier
/KEYS:(abm.n)	See qualifier
/PROCESS:process-type	PROCESS:RECORD
/RELATIVE	
/SEQUENTIAL	/SEQUENTIAL
/SIZE:n	See qualifier
/SPECIFICATION:file-spec	See qualifier
/OUTPUT:file-spec	See qualifier

Input File Qualifier	
/FORMAT:type	Required
/INDEXED:n	

Prompt:

FILE? input-file-spec [/file-qualifiers]

Command Parameters:

input-file-spec	Specifies the file whose contents are to be sorted. If no file type is given, .DAT is the default file type.

Command Qualifiers:

/ALLOCATION:n	Specifies the initial disk space allocation for the output file. Legal values range from 0 to 65535 (bytes). Output file allocation defaults to the input file size.
/BLOCKSIZE:n	Specifies the blocksize in bytes for any magnetic tape files that may be involved in the sort. This qualifier is valid for magnetic tapes only. The default is a 512 byte block.

/BUCKETSIZE:n

Specifies the number of 512 byte blocks per bucket in a disk output file. If this qualifier is used, the block size is 512 bytes, regardless of any /BLOCKSIZE specification. If the input and output files have the same organization, the output file defaults to the same bucketsize as for the input file. Otherwise the default bucket size is 1.

/[NO] CONTIGUOUS

Specifies whether the disk output file allocation must be contiguous or not. In a contiguous file, each successive block is physically located between its logical predecessor and its logical successor with no filler or extraneous material separating the blocks. /NOCONTIGUOUS is the default.

/DEVICE:device-name

Specifies the device to be associated with the intermediate scratch files of the sort. This qualifier overrides device specifications for scratch files resulting from task build options.

See also the description of the /FILES qualifier that follows, and refer to the TRAX SORT Reference Manual for detailed information about scratch files and their use.

/FILES

Specifies the maximum number of intermediate scratch files. Default is 5. See the TRAX SORT Reference Manual for detailed information on scratch files.

/FORMAT:type:n

Specifies the record format and maximum record size of a file. If /FORMAT appears in the command prior to the input file specification, as a command qualifier, it qualifies the output file specification only.

The /FORMAT qualifier must always be present in the command as an input file qualifier. If /FORMAT is omitted as an output file qualifier, the /FORMAT for the input file applies also to the output file.

The type argument can be any of the following:

FIXED
STREAM
VARIABLE
UNKNOWN

The record size n is the exact record size in bytes for FIXED records, and the maximum record size in bytes for other record formats. Record size may be omitted for output files.

The output record format defaults to that of the input file.

/KEYS:(abm.n, . . .)

Specifies the key fields to control the record sequence of the output file. Up to 10 key fields, separated by commas, are allowed. The entire list of key descriptions must be enclosed in parentheses.

Each field description sequence abm.n breaks down as follows:

1. Specifies how the data shall be treated. Legal values of and their interpretations are:

   B two's complement binary
   C alphanumeric (this is the default)
   D One of the following:
   a. if the characters are alphabetic, numeric with the sign superimposed over the units digit, or certain slashes (/), use the value of the digits group. Here are two examples and their values:

      A2CD5 = (+) 12345 A/47J = (-) 11471

   b. if the characters represent a standard FORTRAN IV number, such as 12, -35, 42.98 or -0.76E+3, convert the number to binary for storage or evaluation
   F 1- or 4-word floating point binary
   I same as D, but with the sign leading and separate, so that the first byte of the field is a + or -
   J same as I but with the sign trailing and separate
   K same as D but with the sign leading and over-punched (54321, for instance, if positive, would come out as 5432A. The negative 54321 would be 5432J.)
   P packed decimal format
   Z ASCII zone

2. Defines the general sort order. The default is N (ascending order).

   N ascending order
   O opposite or descending order

m is a decimal number giving the first byte of the key field. Number from the first byte of the record which is byte 1. This item must be present.

n is a decimal number giving the length of the key field in bytes. This item must be present.

The default abm.n value is

a = C
b = N
m = first position of the field
n = length of field

/OUTPUT:file-spec

Specifies the file that will receive the sorted records. Default is the input file.

/PROCESS:process-type

Specifies which sorting process shall be used. This qualifier is illegal when the /SPECIFICATION qualifier is present. The process-type argument has two possible values:

RECORD
TAG

RECORD specifies a record sort. It produces a reordered file by directly transferring the entire record contents on examination of the record keys. This is is the default.

TAG specifies a tag sort. It produces a reordered file in two stages. First, the key fields from the various records are sorted and given record pointers. Then the sorted file is created by using the sorted record pointers to access the input file records and create the full sorted file.

/RELATIVE

Specifies the organization of the output file. /SEQUENTIAL is the default.

/SIZE:n

Specifies the size of the retrieval window. The value n corresponds to the pack default set up by the /WINDOW qualifier on the INITIALIZE or MOUNT command.

/SPECIFICATION:file-spec

Specifies a file containing a set of controls for the sorting process. The /SPECIFICATION qualifier takes the place of /KEY and /PROCESS qualifiers, and offers greater flexibility in sorting files of non-uniform format.

The specification file includes the following controls:

Record selection
Alternative collating sequence
Forced keys
Variable input format
Variable output format
Process selection

The detailed description of the specification file is beyond the scope of this manual. See the TRAX SORT Reference Manual for further information.

Input File Qualifiers:

/FORMAT:type:n

Specifies the record format and maximum record size for the input file. This file qualifier is required.

The type argument can be any of the following:

FIXED
VARIABLE
UNKNOWN

The n argument gives the exact record size in bytes for FIXED files, on the maximum record size for other record formats.

/INDEXED:n

Specifies indexed sequential file organization for the input file, and gives the number of access keys, n, defined for that file.

Notes:

1. Either a /KEYS or /SPECIFICATION is required in the SORT command.

2. The qualifiers /KEYS or /PROCESS must not be used if /SPECIFICATION is used.

Example:

The file DATA.DAT consists of variable length records, with no record longer than 80 bytes. A RECORD sort is performed, because /PROCESS=RECORD is the default. The sort key begins in record position 1 and is 4 bytes long. An alphanumeric ascending sort is performed. The output is placed in the next higher version of DATA.DAT.

```
>SORT/KEYS:(1,4) DATA/FORMAT:VARIABLE:80
```

9.40 SUBMIT

The SUBMIT command accumulates one or more specified batch command files into a job and places the job in a specified batch queue.

Format:

SUBMIT [/qualifiers] file-spec [, . . .]

Command Qualifiers:	*Defaults:*
/QUEUE:queue-name	/QUEUE:BATCH
/PRIORITY:n	n:50
/[NO] RESTART	/RESTART
/[NO] ORIGINAL	/NOORIGINAL
/[NO] PRINT	/PRINT
/JOB: jobname	First six characters of first file-spec
/AFTER: (dd-mmm-yy hh:mm)	Current time

Prompt:

FILE? file-spec [, . . .]

Command Parameter:

file-spec	Specifies the file containing batch commands. If no file type is included, .CMD is the default.

Command Qualifiers:

/QUEUE:queue-name	Specifies the batch queue into which the job is to be placed. The default queue-name is BATCH
	/PRIORITY:n
	Specifies the queue priority of the job.
/[NO] RESTART	Indicates whether or not the job can be restarted from the beginning in the event that it is interrupted for some reason.
/[NO] PRINT	Specifies whether or not to print the log file for the job.

/[NO] ORIGINAL	Specifies whether or not the system should make temporary copies of files to be submitted from a private volume. /ORIGINAL indicates no temporary copies are to be made; i.e., the original copy will be submitted to the batch or print queue. This allows private volumes to be dismounted.
/JOB: job-name	Specifies a name for the batch job.
/AFTER: (dd-mmm-yy hh:mm)	Specifies a date and time after which the job shall be made eligible for submission to a batch processor.

Examples:

This command places a batch job, name BATCH1 and containing the file BATCH1.CMD into the queue named BATCH'

```
>SUBMIT
FILE? BATCH1
>
```

The job TEST shall be placed in the queue BAT and become eligible for processing after 5:30 p.m. on January 30, 1978. The files FIRST.CTL and SECOND.CTL contain the batch commands of which TEST will consist.

```
>SUBMIT/QUEUE:BATCH/AFTER:(30-JAN-78 17:30)-

DCL>/JOB:TEST -

DCL>FIRST.CTL, SECOND.CTL
>
```

9.41 TYPE

The TYPE command prints or displays the contents of one or more specified files on the issuing terminal.

Format:

TYPE file-spec [, . . .]

Prompts:

FILE? file-spec [, . . .]

Command Parameter:

file-spec	Specifies a file to be printed or displayed on the terminal or in the batch log file.

Command Qualifiers: None.

Note:
Each file-spec must include a file name and a file type. Wildcards are permitted in the file name, file type, and file version components of the file specification.

Example:

The following command displays the contents of the file A.CBL on the terminal from which the TYPE command is issued.

```
>TYPE A.CBL
IDENTIFICATION DIVISION.
PROGRAM-ID.  SAMPLE.
REMARKS.  THIS JUST PRINTS A BRIEF MESSAGE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. PDP-11-70.
OBJECT-COMPUTER. PDP-11-70.
DATA DIVISION.
WORKING-STORAGE SECTION.
01      END-MESSAGE PIC X(40) VALUE IS "THE TASK IS COMPLETED.".
PROCEDURE DIVISION.
MESSAGE-PRINT.
        DISPLAY END-MESSAGE.
        STOP RUN.
>
```

9.42 UNLOCK

The UNLOCK command releases a locked file for access.

Format:

UNLOCK file-spec [, . . .]

Prompt:

FILE? file-spec [, . . .]

Command Parameter:

file-spec	Specifies a file to be released from a locked condition for access. You must specify both a file name and a file type.

Command Qualifier: None.

Notes:

1. Locked files occur as the result of being stored by the system under abnormal conditions. If they are open during an ABORT operation, for instance, they are stored without being normally closed. Locked files may not contain the information you expect, due to the abnormal closing.

2. Locked files are indicated in the directory listing by an L between the block count and the storage date.

Example:

The following sequence demonstrates the use of an UNLOCK command to gain access to a locked file.

```
>COBOL A.CBL

DCL>ABORT COBOL
>
14:56:25    TASK "COBT4 " TERMINATED
            ABORTED VIA DIRECTIVE OR MCR
>DIRECTORY A.*


DIRECTORY DB0:[350,230]
19-JUL-78 14:56

A.TST;1              0.          02-JUN-78 13:56
A.LST;1              1.          02-JUN-78 13:56
A.ODL;1              1.          18-JUL-78 14:01
A.OBJ;1              2.          18-JUL-78 14:01
A.TSK;1             27.      C  18-JUL-78 14:01
A.CBL;3              1.          18-JUL-78 14:03
A.TMP;2              0.        L 19-JUL-78 14:56

TOTAL OF 32./32. BLOCKS IN 7. FILES


>UNLOCK A.TMP
>DIRECTORY A.TMP
```

```
DIRECTORY DB0:[350,230]
19-JUL-78 14:59

A.TMP;2              0.          19-JUL-78 14:56

TOTAL OF 0./0. BLOCKS IN 1. FILE


>
```

APPENDIX A

THE RMSDEF INTERACTIVE UTILITY

A.1 PURPOSE

The interactive RMSDEF utility creates RMS files, allowing you to control all attributes of the files.

A.2 EFFECT

You specify file attributes by responding to requests for data and questions from the utility. The method of questioning is outlined in Figure A-1. The figure shows the general flow of processing. You may also get help from the utility by typing a question mark (?) in response to any question or request for data.

NOTE

The flowchart in Figure A-1 contains circles with section numbers in them. These section numbers flag the different areas of the utility's processing. Each numbered section of text expands and explains the associated portion of the flowchart.

RMSDEF also has the capability of building an indirect command file while you operate it. This command file may be used thereafter to create file(s) and may be modified to create other similar files. See Sections A.3 and A.4.1.

RMSDEF, however, does not write records into the file. The creation of the data contents of the file must occur after the RMSDEF utility has created the file. You can employ either an application program or the MERGE command to write records into the file.

The following list indicates the information that RMSDEF will always request as well as the requests that may be made depending on specifications already typed.

1. Command File?
   a. If yes, file specification
   b. If yes, create only command file or create both RMS and command file?
2. File Specification
3. Data Structure
   a. Minimum
      (1) File organization
      (2) Record format

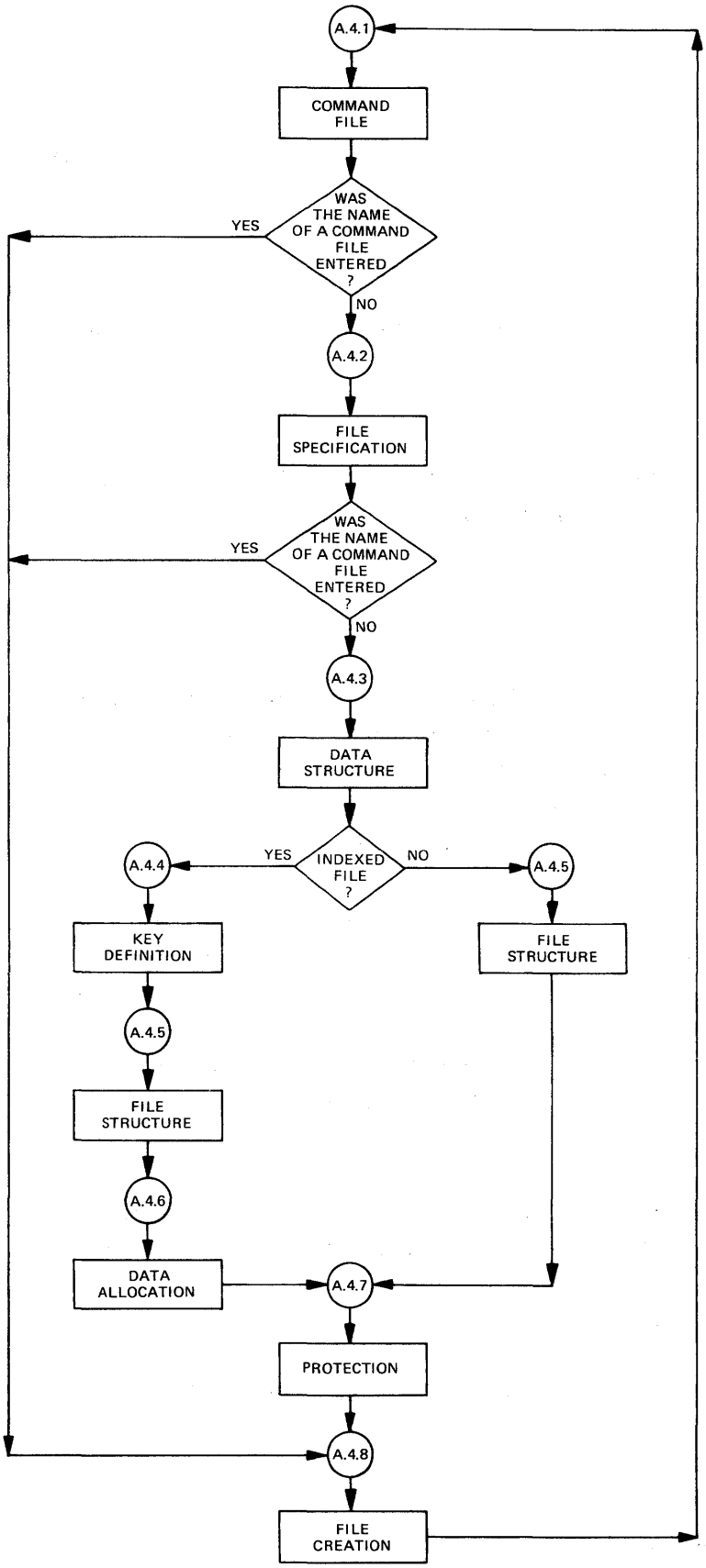

Figure A-1 Interactive DEFINE Processing

(3) Maximum record size
(4) CARRIAGE RETURN control?

b. Possible
(1) Size of fixed control area for VFC records
(2) Maximum number of records in a relative file
(3) Block-spanning records in a sequential file?
(4) FORTRAN character control if no CARRIAGE RETURN control?

4. Key Definition (indexed files only)
a. Minimum
(1) Position of key
(2) Size of key
(3) Data type
(4) Name of key
(5) Duplicate keys?
(6) Null key value?

b. Possible
(1) Change keys if duplicatable?
(2) Null key character if null key value

5. File Structure
a. Minimum
(1) Areas? (indexed files only)
(2) Placement control?
(3) Initial allocation quantity
(4) Default extension quantity
(5) Contiguous?

b. Possible
(1) Location if placement control
(2) Exactly if placement control?
(3) Type of alignment, if placement control, and areas

6. Data Allocation (indexed files only)
a. Minimum
(1) Number of bytes in data buckets filled
(2) Number of bytes in index buckets filled

b. Possible
(1) Area containing index level 0 for each key if areas
(2) Area containing index levels 2+ for each key if areas
(3) Area containing index level 1 if areas

7. Protection

A.3 UTILITY CALL AND TERMINATION

Call the RMSDEF utility with the following command:

```
RUN $RMSDEF
```

The utility prints:

DO YOU WANT TO GENERATE A COMMAND FILE FOR FUTURE USE(NO)?

See Section A.4 for the complete dialog sequence.

You may terminate RMSDEF at any time by typing a CTRL/Z. Control is passed back to DCL.

A.4 PROCESS

A.4.1 Command File

1. The terminal prints:
   DO YOU WANT TO GENERATE A COMMAND FILE FOR FUTURE USE(NO)?
   Type one of the following:
   - Y — If you want to enter a filespec for an indirect command file and have RMSDEF write entries into the file as you move through the utility. Go to step 2.
   - N or (RET) — If you do not want to build a command file. Go to Section A.4.2, File Specification.
   - filespec — If you have already built a command file with RMSDEF and want RMSDEF to read it now and create the specified file. Go to Section A.4.8, File Creation.
   - ID — If you want the utility to identify itself with a version number. RMSDEF prints the following message:
     THIS IS THE RMS RMSDEF UTILITY, VERSION n
     where n is the revision level of the utility itself.
2. The terminal prints:
   ENTER A FILE SPECIFICATION FOR THE INDIRECT FILE YOU WANT:
   Type a filespec.
3. The terminal prints:
   DO YOU WANT TO CREATE THE FILE YOU WILL BE DESCRIBING(NO)?
   Type one of the following:
   - Y — If you want RMSDEF to create the file specified as well as the command file.
   - N or (RET) — If you do not want to create the RMS file, only the command file that may be used to create the file at a later date.
4. The command file created by RMSDEF takes the following form. The utility follows each comment with the appropriate sequence of your entries, each on a separate line. Where the user enters only (RET) to accept the default value, RMSDEF places CR/LF on a separate line in the command file.

:THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION
:THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTRIBUTES
:THE FOLLOWING QUESTIONS DEAL WITH KEYS (for Indexed files only)
:THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
:THE NEXT QUESTIONS ASK ABOUT FILL SIZES FOR KEYS
:THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION

A.4.2 File Specification

The terminal prints:
ENTER YOUR FILE SPECIFICATION:
Type one of the following:

filespec	If you want to create (or simulate creation, see Section A.4.1, Command File, step 3) an RMS file. Go to Section A.4.3, Data Structure.
@filespec	If you have built a command file (see Section A.4.1, Command File) and want RMSDEF to use it to create the file specified. Go to Section A.4.8, File Creation.

A.4.3 Data Structure

1. The terminal prints:
   FILE ORGANIZATION (SEQ):
   Type one of the following:
   SEQ or (RET) for sequential organization
   REL for relative organization
   IDX for indexed organization

   NOTE

   If you indicated a magnetic tape device in the filespec, RMSDEF does not request a file organization. Since a magtape file requires sequential organization, the utility prints:

   SINCE YOU SPECIFIED A NON-DISK DEVICE,
   YOUR FILE ORGANIZATION MUST BE
   SEQUENTIAL

2. The terminal prints:
   RECORD FORMAT (VAR):
   Type one of the following:

VAR or (RET)	If the records in the file will have differing, or variable, lengths.
FIX	If the records in the file will have the same, or a fixed, length.

VFC	If each record in the file will have a control area with a fixed length and a data area of no standard length; that is, variable with fixed control.
STM	If the records in the file will have no specific format but are delimited only by record terminator characters. This stream format is permitted for sequential disk files only.
UDF	If there are no records (or you don't want RMS to recognize records) in the file; this format is used only for block I/O files, such as RMS backup files.

RMSDEF will reject a format that is illegal with the file organization already specified; for instance, STM for indexed files.

3. If you specified VFC in step 2, the terminal prints the following message; otherwise, go to step 4.

   SIZE OF FIXED CONTROL AREA(2):

   Type the decimal number of bytes in the fixed control area of each record in the file. The minimum size is one byte; the maximum size is 255 bytes; the default is two bytes.

4. The terminal prints:

   MAXIMUM RECORD SIZE: or MAXIMUM RECORD SIZE (O):

   Type a decimal number indicating the maximum number of bytes in any record in the file. RMS checks this value whenever a record access operation is requested for this file: if the record specified exceeds the maximum size, RMS returns an error. A size of zero disables the RMS check, but a nonzero value is required for all relative files and all files with fixed-length records.

5. If you specified REL in step 1, the terminal prints the following message; otherwise, go to step 6.

   MAXIMUM NUMBER OF RECORDS (0):

   Type a decimal number indicating the maximum number of records that this relative file will contain. RMS checks this value whenever a record access operation is requested for this file: if the relative record number specified exceeds the maximum record number, RMS returns an error. (RET) sets the number to zero, which disables the RMS check. The zero allows the file to contain as many records as is physically possible (the technical maximum is 2.14748×10^{-9}).

6. If you specified SEQ in step 1, the terminal prints the following message; otherwise, go to step 7.

   WILL YOU ALLOW RECORDS TO CROSS BLOCK BOUNDARIES (YES)?

   Type one of the following:

Y or (RET)	If you want records to cross block boundaries.
N	If you do not want records to span blocks. If you specified FIX in step 2 and a maximum record size greater than 512 in step 4, the terminal prints:

SINCE YOU SPECIFIED FIXED SIZE RECORDS, YOU MUST HAVE A MAXIMUM RECORD SIZE LESS THAN 512. (THE SIZE OF 1 BLOCK) OR YOU MUST ALLOW

RECORDS TO CROSS BLOCK BOUNDARIES. HERE'S YOUR CHANCE TO CHANGE 1 OF THESE.

RMSDEF repeats steps 4 and 6. Change either your MRS or the answer to crossing block boundaries.

7. The terminal prints:
   DO YOU WANT CARRIAGE RETURN CONTROL (YES)?
   Type one of the following:

   Y or (RET) — If you want each record to be preceded by a line feed character and followed by a carriage return character when it is written to a carriage control device (printer, terminal, and so on). See the note below and go to appropriate section.

   N — If you do not want carriage return control and/or you do want FORTRAN character control. Go to step 8.

8. The terminal prints:
   DO YOU WANT FORTRAN CHARACTER CONTROL (NO)?
   Type one of the following:

   Y — If you want the first byte of each record to be allocated for a FORTRAN forms control character.

   N or (RET) — If you do not want FORTRAN character control.

NOTE

If you indicated a magtape device in the file specification, at this point RMSDEF requests:

MAGTAPE BLOCK SIZE (512):

Type a decimal number between 18 and 8192 representing the number of bytes in each tape block. The number should be a multiple of four; if it is not, RMS will round it up to the next multiple of four before writing it as an attribute. A (RET) sets the size to the default of 512 bytes.

The utility then bypasses all other processing and immediately requests protection information (see Section A.4.7).

A.4.4 Key Definition

As indicated by Figure A-1, this section applies only to indexed files.

1. The terminal prints:
   IT'S TIME TO DEFINE THE PRIMARY KEY
   POSITION OF KEY:
   Type a decimal number indicating the position of the first byte of the key within each record. For instance, if the key starts with the first byte of the record, its position is 0. The second byte has position 1 and so on.

   A position number must be specified for each segment of a segmented key; the numbers are separated by commas and enclosed in parentheses.

2. The terminal prints:
   SIZE OF KEY
   Type the decimal number of bytes in the key; that is, its length. Minimum length is 1 byte; maximum is 255 bytes; there is no default.

   A length must be specified for each segment of a segmented key; the numbers are separated by commas and enclosed in parentheses. A length must be typed for each position number specified in step 1, but the sum of all lengths cannot exceed 255.
3. The terminal prints:
   DATA TYPE(STRING):
   Type one of the following:

STR or (RET)	If your key value will be a string of alphanumeric characters.
IN2	If your key value is a 15-bit signed integer.
IN4	If your key value is a 31-bit signed integer.
BN2	If your key value is a 16-bit unsigned binary number.
BN4	If your key value is a 32-bit unsigned binary number.
PKD	If your key value is a packed decimal number.
Y or (RET)	

4. The terminal prints:
   ENTER A NAME FOR YOUR KEY, IF YOU SO DESIRE(NONE):
   Type one of the following:

(RET)	If you do not want to specify a name for the key.
name	If you want to name the key being defined; up to 32 ASCII characters are allowed.

5. The terminal prints:
   WILL YOU ALLOW DUPLICATE KEYS(dflt)?
   Type one of the following:

Y	If the file may contain more than one record with the same value for this key. Keys must be specified as duplicatable before they can be specified as changeable.
N	If each record in the file must have a unique value for this key. RMS returns an error if duplication is attempted; that is, a write or update operation will fail for a record that has a value in this key field exactly like a record already in the file.

   Defaults and the values of dflt are:
   Primary key - NO
   Alternate keys - YES

NOTE

Steps 5, 6, and 7 apply only to alternate keys.

6. If you specified YES in step 5, the terminal prints the following message; otherwise, go to step 7.
   WILL YOU ALLOW KEYS TO CHANGE(YES)?
   Type one of the following:

Y or (RET) — If this alternate key may be changed during an update operation; that is, the record may be read with one value for the key and rewritten with another value for the same key.

N — If this alternate key must not change after the record is originally created.

7. The terminal prints:
   DO YOU WISH TO DEFINE A NULL KEY VALUE(NO)?
   Type one of the following:

   Y — If you want the file to contain some records that cannot be accessed via this key. When RMS writes a record into an indexed file, it normally updates all indexes of the file to reflect the values found in the corresponding key fields of the record. However, if a null key value is defined for an alternate key, RMS examines the contents of the key field in the record. If this field consists solely of the null key character specified, RMS will not make an entry in the associated alternate index for that particular record. Go to step 7.

   N or (RET) — If you do not want to specify a null value for this key. Go to step 8.

8. The terminal prints:
   O.K., ENTER YOUR NULL KEY VALUE CHARACTER:
   Type one of the following:

   c — The single character itself, if it is not #, ?, or @.

   #43 — For the reserved character #.

   #77 — For the reserved character ?.

   #100 — For the reserved character @.

   #n — Any octal byte value (000-377) specified by n.

9. The terminal prints:
   DO YOU WANT TO DEFINE MORE KEYS(NO)?
   Type one of the following:

   Y — If you want to define more keys for the file. You may define up to 254 alternate keys; however:

   - The MERGE command will not read higher than the ninth alternate key; that is, the NUMBER: n option of the MERGE command must be less than or equal to nine.
   - Your application language may not support that many keys. See the appropriate user's guide.

   RMSDEF prints:

   ALTERNATE KEY n

   where n starts with 1 and is incremented each time you answer Y.

   RMSDEF then requests this information for each alternate key indicated; the alternate keys are defined in order, beginning with the first alternate after the primary key has been defined.

   N or (RET) — If all keys for this file have been defined.

A.4.5 File Structure

1. If you specified IDX for file organization, the terminal prints the following message; otherwise, go to step 2.
   DO YOU WANT TO DEFINE AREAS(NO)?
   Type one of the following:
   - Y — If you want parts of this file to be logically different, with different attributes. The questions 2-10 will be asked for each area.
   - N or (RET) — If you want this file located in one area. RMSDEF asks you to go to step 2.
2. The terminal prints:
   DO YOU WANT PLACEMENT CONTROL (NO)?
   Type one of the following:
   - Y — If you want to specify an exact location on disk for this area. Go to step 3.
   - N or (RET) — If you do not want to specifically locate this area. Go to step 6.
3. If at least one area has already been defined; that is, you answered YES in step 1 at least once, the terminal prints the following message. Otherwise, go to step 4.
   WHAT TYPE OF ALIGNMENT DO YOU WANT (LBN)?
   Type one of the following:
   - LBN or (RET) — If the location you will specify in step 4 is a Logical Block Number (LBN) on the disk volume.
   - VBN — If the location you will specify in step 4 is a Virtual Block Number (VBN) already established within the file itself; that is, in a previously defined area.
4. The terminal prints:
   LOCATION:
   Type the decimal number location of the first block for this file area.
5. The terminal prints:
   EXACTLY (NO)?
   Type one of the following:
   - N or (RET) — If you will accept the closest approximation of the LBN or VBN location specified in step 4, if the exact location is not available.
   - Y — If this area must start in the exact LBN location specified in step 4. If this location is not available, RMSDEF will print an error message when it tries to create the file and give you another chance to reconsider this question. Exact VBN locations are already taken, by definition.
6. The terminal prints:
   ALLOCATION (0 – IT IS SUGGESTED YOU ENTER A VALUE):
   Type a decimal number indicating the initial size of the area in blocks. A (RET) sets the value to zero: the area will be created, but it will have to be expanded before any records can be written into it. Since automatic file extension is a time-consuming procedure, the file should be fully allocated when it is created.
7. If you specified REL or IDX for file organization, the terminal prints the following message; otherwise, go to step 8.
   BUCKET SIZE(1):

Type a decimal number indicating the number of blocks in a bucket for this area. The minimum is the number of blocks that will contain one record (according to the size specified in Section A.4.3, step 4); the maximum is 32 blocks; the default is one. This number determines the number of blocks read into memory during each file access operation and therefore affects speed of processing and the amount of memory a program accessing this file requires.

8. The terminal prints:
   DEFAULT EXTENSION QUANTITY (0 – IT IS SUGGESTED YOU ENTER A VALUE):
   Type a decimal number indicating the number of blocks that should be added to the area each time RMS must extend it. The Default Extension Quantity (DEQ) should be a multiple of the bucket size. RMS requests this number of blocks from the operating system.

   A carriage return sets the value to zero: RMS will add only the minimum amount of space required each time it expands the area. A definite but reasonable extension quantity speeds up processing.
9. The terminal prints:
   DO YOU WANT A CONTIGUOUS AREA (NO)?
   or
   DO YOU WANT A CONTIGUOUS FILE (NO)?
   Type one of the following:

Y (RET)	If you want the disk space for this area allocated in contiguous, that is, physically adjoining, blocks. If RMS cannot find that much contiguous space, it will not create the file, even if sufficient non-contiguous blocks are available. A contiguous file or area may be extended although the disk space added will probably not be contiguous with the original allocation.
N or (RET)	If you do not require contiguous block allocation.

10. If you answered YES in step 1 (you have an indexed file), the terminal prints the following messages; otherwise, go to the next appropriate section.
    JUST FINISHED WITH AREA NUMBER n
    DO YOU WANT TO DEFINE MORE AREAS (NO)?
    Type one of the following:

Y	If you want to specify attributes for another area of your file. Areas are numbered sequentially, starting with zero. The areas will be associated with the index and data portions of the file in the next section of the utility. Go to step 2.
N or (RET)	If you have defined enough areas for this file. Go to step 11.

11. If you defined one or more areas with a Default Extension Quantity (DEQ) of zero, the terminal prints the following message; otherwise, go to the next section.
    DEFAULT EXTEND QUANTITY FOR YOUR FILE (0):
    Type a decimal number indicating the number of blocks that should be added to the file each time RMS must extend it. The DEQ should be a multiple of the bucket size. A (RET) sets the value to zero: RMS will add only the minimum amount of space required each time it expands the file. A definite but reasonable extension quantity speeds up processing.

A.4.6 Data Allocation

As indicated by Figure A-1, this section applies only to indexed files. RMSDEF begins this portion of dialogue with the message:

IT IS TIME FOR AREA NUMBERS AND FILL FACTORS FOR KEYS.

The questions are asked for each key defined (see Section A.4.4, Key Definition).

1. The terminal prints:
   AREA NUMBER FOR DATA BUCKETS FOR THIS KEY (0):
   Type an integer (0-n) indicating the area already defined (see Section A.4.4, Key Definition) which should contain the data portion (Level 0) of this key.
2. The terminal prints:
   THE BUCKET SIZE IS nnn
   HOW MANY BYTES DO YOU WANT FILLED IN THE DATA BUCKET (0)?
   Type a decimal number of bytes in each of this key's data buckets that should be used during the original population of the file. This number is honored by the MERGE command and may be honored by MACRO programs.
   A (RET) sets the number to zero, indicating that buckets will be completely filled and that no free space will be available for records added during update operations.
3. The terminal prints:
   AREA NUMBER FOR INDEX BUCKETS FOR THIS KEY (0):
   Type an integer (0-n) indicating the area already defined (see Section A.4.4, Key Definition) which should contain the upper portions (Levels 2+) of the index for this key.
4. The terminal prints:
   THE BUCKET SIZE IS nnn
   HOW MANY BYTES DO YOU WANT FILLED IN THE INDEX BUCKET (0)?
   Type a decimal number of bytes in each of this key's index buckets that should be used during the original population of the file.
   A (RET) sets the number to zero, indicating that buckets will be completely filled and that no free space will be available for records added during update operations.
5. The terminal prints:
   AREA NUMBER FOR THE LOWEST INDEX LEVEL FOR THIS KEY (0):
   Type an integer (0-n) indicating the area already defined (Section A.4.4, Key Definition) which should contain the lowest index level portion of this key.
   If the area you specified for the upper portions of the index (Levels 2+) and the area you specified here for Level 1 have different bucket sizes, RMSDEF prints the following message and returns to step 3:
   THE AREA ASSOCIATED WITH THE LOWEST LEVEL INDEX BUCKET HAS A DIFFERENT BUCKET SIZE THAN THE AREA ASSOCIATED WITH THE HIGHER INDEX BUCKET. TRY BOTH AGAIN.

A.4.7 Protection

1. The terminal prints:
   SPECIFY PROTECTION BY CLASS:
   OWNER: (RWED ALLOWED)

Type one of the following:

RET	If you want this file completely available to access (Read, Write, Edit, Delete) by the account current when the file is created.
NONE	If you do not want the file owner to have any access to this file after it is created.
R, W, E, and/or D	To specify a level of protection between none and all. One or more of the letters representing Read, Write, Edit, and Delete may be specified, in that order and without separation.

2. The terminal prints:
   GROUP: (RWED ALLOWED)
   Type one of the following:

RET	If you want this file completely available to access (Read, Write, Edit, Delete) by all accounts with the same group number as the owner's account.
NONE	If you do not want the group members to have any access to this file after it is created.
R, W, E, and/or D	If you want to specify a level of protection between none and all. One or more of the letters representing Read, Write, Edit, and Delete may be specified, in that order and without separation.

3. The terminal prints:
   SYSTEM: (RWED ALLOWED)
   Type one of the following:

RET	If you want this file completely available to access (Read, Write, Edit, Delete) by system privileged accounts.
NONE	If you do not want privileged accounts to have any access to this file after it is created.
R, W, E, and/or D	If you want to specify a level of protection between none and all. One or more of the letters representing Read, Write, Edit, and Delete may be specified, in that order and without separation.

4. The terminal prints:
   WORLD: (R ALLOWED)
   Type one of the following:

RET	If you want this file to have Read access only for all accounts, including those outside the owner, group, and privileged accounts.
NONE	If you do not want other accounts to have any access to this file after it is created.
R, W, E, and/or D	If you want to specify a level of protection other than Read. One or more of the letters representing Read, Write, Edit, and Delete may be specified in that order and without separation.

A.4.8 File Creation

The RMSDEF utility attempts to create the file.

A.4.8.1 Success - If RMS does not return an error, the utility prints:

YOUR FILE HAS BEEN CREATED!! – filespec

If you chose to create a command file (see Section A.4.1, Command File), RMSDEF also prints:

YOUR FILE HAS BEEN PROCESSED AND A COMMAND FILE GENERATED!! – filespec
DO YOU WANT TO CLOSE THE INDIRECT FILE (NO)?

Type one of the following:

Y	If you are finished specifying files for the command file. The utility returns to the question about command file generation (see Section A.4.1, Command File).
N or RET	If you want to specify another RMS file and you want the command file to include your input. RMSDEF continues to use the command file originally specified and to obey your answer to the DO YOU WANT TO CREATE THE FILE YOU WILL BE DESCRIBING? question (see Section A.4.1, step 3). The utility returns to the request for a file specification (see Section A.4.2, File Specification).

You may start the process to create another file or type CTRL/Z (^Z) to terminate the utility.

A.4.8.2 Error - RMSDEF allows three types of recoverable creation errors, shown next. All other errors result in a description of the error and a message that the file as specified cannot be defined. The utility returns to the file specification request to let you try again (see Section A.4.2).

1. The terminal prints an error description, followed by:
   THIS FILE CANNOT BE CREATED DUE TO AN ERROR IN THE FILE SPECIFICATION. DO YOU WISH TO REENTER THE ENTIRE FILE SPECIFICATION (YES)?
   Type one of the following:

Y or RET	If you know how to correct the error and/or want to enter another filespec. The utility requests the filespec and attempts to create the file using it.
N	If you don't know how to correct the error and/or want to start again. RMSDEF returns to the file specification request.

2. The terminal prints:
   THE FILE WASN'T CREATED SINCE YOU SPECIFIED A BLOCK WHICH IS IN USE. WILL YOU NOW ACCEPT AN APPROXIMATION OF THAT LOCATION (YES)?
   You requested an exact placement of a file or area (Section A.4.5, File Structure, step 5).
   Type one of the following:

Y or RET	If you now want the best approximation of the location you specified for your file or one or more areas in your file. RMSDEF will try to create the file again.

N	If you want the exact location or nothing. RMSDEF returns to the file specification request since it cannot create the file as specified.

3. The terminal prints:
   A FILE WITH THE FILE SPECIFICATION YOU ENTERED ALREADY EXISTS.
   DO YOU WANT TO SUPERSEDE THE FILE (NO)?
   Type one of the following:

Y	If you want to delete the file that already exists and create the file you have specified through the utility. RMSDEF deletes the existing file and attempts to create the specified one.
N or RET	If you do not want to supersede the existing file with the one you have just specified. RMSDEF returns to the file specification request.

APPENDIX B
TRAX SUPPORT ENVIRONMENT MESSAGES

This appendix describes the system messages created by the TRAX Support Environment commands. All commands that can be issued from the TRAX Support Environment with the exception of the BASIC, COBOL, and MACRO programming languages and the TRAX Editor error messages are listed. These error messages are described in their respective user reference manuals.

B.1 ABORT

These are the error messages created by the ABORT command.

ABO – TASK MARKED FOR ABORT

An attempt has been made to abort a task which is already marked for abort.

ABO – TASK NOT ACTIVE

The specified task is not currently active.

Messages from Task Termination Notification Routine (TKTN):

TKTN displays information about task aborts, whether caused by an explicit ABORT command or some other force. The display has the format:

TASK "<taskname>" TERMINATED
<abort cause>

Following the displayed cause for the abort is a list of the task's registers at the time of the abort. The possible causes of the abort are described below.

Abort Cause Messages:

ABORTED BY DIRECTIVE OR MCR

Either TRAX or an Executive directive issued by another task caused the task to be aborted.

ABORTED VIA MCR

TRAX aborted the task and requested a post-mortem dump.

CHECKPOINT FAILURE. READ ERROR.

The task could not be read back into memory from disk after being checkpointed.

LOAD FAILURE. READ ERROR

The task could not be loaded from disk because of a hardware error.

PARITY ERROR

A parity error occurred while the task was executing. The task was fixed in memory so that the memory could not be reused by another task.

TASK EXIT WITH OUTSTANDING IO

The task exited with one or more outstanding I/O requests. Tasks should terminate all I/O operations before exiting. The system does, however, clean up all outstanding I/O.

B.2 ALLOCATE

These are the error messages created by the ALLOCATE command.

ALL – DEVICE ATTACHED

The specified device cannot be allocated because it is attached to a running task.

ALL – PSEUDO DEVICE ERROR

The specified device is a pseudo device. Pseudo devices cannot be allocated.

ALL – PUBLIC DEVICE

The command attempted to allocated a public device. Public devices cannot be allocated.

ALL – USER LOGGED ON TERMINAL

The command attempted to allocate a terminal that has been logged-in by another user. Logged-in terminals cannot be allocated.

B.3 APPEND

These are the error messages created by the APPEND command.

NOTE

A fatal error in the cnv utility is marked by a preceeding question mark "?". If the message has a question mark in brackets [?] the error may be either fatal or diagnostic. If the error message has no preceeding question mark the error is diagnostic. The error messages prefixed with "DSC" refer to volume archiving. The others refer to file archiving.

?cnv – DEVICE OFF LINE - device

Description

The indicated device exists on the system but the attempt to access it has been prohibited for one of the following reasons.

1. The device is not ready.
2. No volume is mounted on the device.
3. The device is currently reserved by another job.
4. The device requires privileges for ownership and the user does not have privilege.
5. The device has been disabled.

Suggested User Action

Determine the nature of the problem and take corrective action.

cnv – DEVICE/FILE IS FULL - device/filename

Description

The utility cannot create an output file on the indicated device because of insufficient space or the indicated file cannot be extended due to insufficient space.

Suggested User Action

Reenter the command using another device for output files or copy the indicated file to another device and retry the command. Optionally, delete unneeded files on the indicated device and reenter the original command line.

?cnv – FILE NOT AVAILABLE - filename

Description

The indicated file is being accessed for exclusive use by another job.

Suggested User Action

Periodically retry the command until the file has been released.

?cnv–ILLEGAL DEVICE – device

Description

The indicated device does not exist.

Suggested User Action

Reenter the command line with a corrected device specification.

?cnv–NO SUCH KEY FOR FILE – value

Description

The specified key of reference value represents a non-existent key in an indexed file.

Suggested User Action

Reenter the command with a correct key of reference value.

?cnv–NOT A DIRECTORY DEVICE – device

Description

The user has issued a directory-oriented command for a device (such as a printer) that does not have directories (accounts).

Suggested User Action

Reenter the command line without specifying an account.

APP – CANNOT FIND DIRECTORY FILE

Description: UFD specified does not exist on this volume.

Suggested User Response: Reenter the command line, specifying the correct UFD or the correct volume.

APP – CANNOT FIND FILE(S)

Description: The file(s) specified in the command were not found in the designated directory.

Suggested User Response: Check the file specifier and reenter the command line.

APP – I/O ERROR ON INPUT FILE

or

APP – I/O ERROR ON OUTPUT FILE

Description: One of the following conditions may exist:

- The device is not on-line.
- The device is not mounted.
- The hardware has failed.
- The volume is full (output only).
- Input file is corrupted.

Suggested User Response: Determine which condition caused the message and correct that condition. Reenter the command line.

APP – NOT A DIRECTORY DEVICE

Description: A directory-oriented command was issued to a device that does not have directories (such as a printer).

Suggested User Response: Reenter the command line without specifying a UFD.

APP – OPEN FAILURE ON INPUT FILE

or

APP – OPEN FAILURE ON OUTPUT FILE

Description: The specified file could not be opened. One of the following conditions may exist:

- The file is protected against access.
- A problem on the physical device (e.g., device cycled down).
- The volume is not mounted.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.

Suggested User Response: Determine which condition caused the message and correct that condition. Reenter the command line.

B.4 ARCHIVE

These are the error messages that are created by the archive command.

NOTE

A question mark "?" preceeding the bck or rst utility error messages indicates a fatal error. A question mark in brackets [?] indicates that the error may be fatal or diagnostic. If no question mark preceeds the error message the error is diagnostic. The error messages prefixed with "DSC" refer to volume archiving. The others refer to file archiving.

DSC – 7 DUP DEV NAME

The same device was entered more than once in the command.

Re-enter the command string with the devices specified only once.

DSC – 9 DEV device: NOT IN SYSTEM

The specified device is not present in the configuration of the operating system being used.

Check the device identifier that was entered in the command string, and retry the command.

DSC – 10 DEV device: NOT files-11

The specified input device is not formatted as a files-11 device.

Check the input device to ensure it is the one desired, and re-enter the command.

DSC – 14 OUTPUT TAPE ON device: IS NOT AT BOT

The specified continuation tape is not at load point.

Remount or reset the tape at load point and re-enter the command.

DSC – 18 TAPE device: NOT ANSI FORMAT

The tape is not in correct format for a DSC operation.

Check the tape and change if necessary.

DSC – 21 TAPE device: A CONTINUATION TAPE

The tape has been mounted out of sequence.

Re-enter the command, specify input tapes in proper order.

DSC – 23 FAILED TO FIND HOME BLOCK device:

A read error occurred when trying to copy from the input disk. Either the disk is bad, the home block is bad, or the disk is not in files-11 format.

Check the disk in question, change disk drives if possible, and re-enter the command.

DSC – 26 I/O ERROR B ON device:

The I/O error indicated by the message that follows explains why the file header on the input device could not be read. The specified file is lost.

Retry the operation after correcting the cause of the error on the input device.

DSC – 27 I/O ERROR B ON device:

The I/O error indicated by the message that follows explains why the file header on the output device could not be read. The specified file is lost.

Retry the operation after correcting the cause of the error on the output device.

DSC – 28 CODE A

The file header for the storage bit map file cannot be read.

The disk is unusable and therefore cannot be copied.

DSC – 29 I/O ERROR C ON device:

The following message explains the error that occurred while reading the specified file.

Retry the operation.

DSC – 30 I/O ERROR D ON device:

A read error, as indicated by the diagnostic message which follows, occurred when reading the name or boot block of the disk.

Retry the operation on a new drive.

DSC – 31 RELATIVE VOLUME X OF SET NOT MOUNTED

The specified tape is not on the system.

Mount the tape and re-enter the command.

DSC – 36 I/O ERROR E ON device: file id

The message that follows explains the I/O error that occurred while reading the specified file header.

Retry the operation.

DSC – 37 INPUT DEVICE device: file id file number NOT PRESENT

The specified file does not have a file header in the index file; the file is not copied.

This is a warning only. If desired, the operation may be retried on a different disk drive.

DSC – 38 INPUT DEVICE device: file id file number IS DELETED

The specified file was found to be partially deleted on the input disk and was not copied.

This is a warning only. No action is required.

DSC – 39 INPUT DEVICE device: file id UNSUPPORTED STRUCTURE LEVEL

The specified input disk is not a level one (ODS1) disk and cannot be used.

Retry the operation with a level one disk.

DSC – 40 INPUT DEVICE device: file id file number FILE NUMBER CHECK

An incorrect file header was read from disk causing the specified file to be lost.

Retry the operation.

DSC – 41 INPUT DEVICE device: file id file number FILE HEADER CHECKSUM ERROR

Incorrect file header contents cause the specified file to be lost.

Retry the operation.

DSC – 42 INPUT DEVICE device: file id SEQUENCE NUMBER CHECK

The sequence number is incorrect.

Retry the operation and/or replace the disk.

DSC – 43 INPUT DEVICE device: file id file number SEGMENT NUMBER CHECK

The linkage connecting file segments has been broken; the specified file is lost.

Retry the operation.

DSC – 44 DIRECTIVE ERROR

An internal error has occurred, usually the result of a system overload.

Retry the operation.

DSC – 45 I/O ERROR F ON device:

The message that follows indicates that the specified input device may cause a subsequent error.

This message is a warning only. No action is required unless another error message is displayed. If another error message is displayed, correct the cause of the error and re-enter the command.

DSC – 46 I/O ERROR F ON device:

The message that follows indicates that the specified output device may cause a subsequent error.

This message is a warning only. No action is required unless another error message is displayed. If another error message is displayed, correct the cause of the error and re-enter the command.

DSC – 47 I/O ERROR I ON device: file id file number virtual block number

An I/O error occurred which is explained by the message that follows which resulted in bad data being read from the specified virtual block number.

This is a warning message only. The block specified should be examined to determine the extent of the error.

DSC – 48 I/O ERROR I ON device: file id file number virtual block number

An I/O error occurred which is explained by the message that follows which resulted in bad data being read from the specified virtual block number.

This is a warning message only. The block specified should be examined to determine the extent of the error.

DSC – 49 VERIFICATION ERROR ON device: file id virtual block number

This is a warning signifying that the input and output devices did not match.

DSC – 50 BAD DATA BLOCK ON device: file id file number virtual block number

A parity error occurred when copying the blocks contents from disk. The block specified on the output disk contains erroneous data.

When the copy operation is completed, the data contained in the specified block should be examined and corrected.

DSC – 55 INPUT FILE ON device: WILL BE RESYNCHRONIZED

The tape position was lost while reading the input tape. The file specified in the message, as well as some subsequent files, may be lost. Additional error messages will probably occur.

Retry the operation from the beginning.

DSC – 57 OUTPUT FILE HEADER FULL ON device:

Too many blocks on the output disk have caused inconsistencies in file header data. The specified file is lost.

Retry the operation with a different output disk.

DSC – 58 OUTPUT FILE HEADER ON device: NOT MAPPED – file id file number

Space for the specified file header was not allocated. The file is lost.

Retry the operation; a new disk may be required.

DSC – 59 I/O ERROR G ON device:

The message that follows explains the I/O error that occurred while writing the specified file.

Retry the operation.

DSC – 60 FAILED TO READ FILE EXTENSION HEADER ON device: file id file number

When copying from the input disk, an extension header was searched for, but not found. The remainder of the specified file was lost. A problem may exist with the input disk, or a preceding I/O error occurrence may have caused an inconsistency.

Retry the operation.

DSC – 61 FAILED TO ALLOCATE HOME BLOCK device:

The home block cannot be created on the specified disk device because it has too many bad blocks.

Replace the device and re-enter the command.

DSC – 62 INDEX FILE ALLOCATION FAILURE device:

Too many bad blocks exist to allow the allocation for specified file.

Replace the disk and re-enter the command.

DSC – 63 OUTPUT DISK device: IS NOT BOOTABLE

Logical block number 0 of the specified disk or tape is bad.

This is a warning only. No action is required.

DSC – 64 INVALID BAD BLOCK DATA device:

The bad block data on the output disk are invalid.

Run the BAD utility on the disk; manually enter bad block data; or re-enter the command, specifying another disk.

DSC – 65 BAD BLOCK FILE FULL device:

Too many bad blocks exist on the output disk.

Replace the disk and retry the command.

DSC – 66 NO BAD BLOCK DATA FOUND device:

No bad block data exists for the specified output disk.

If bad block data is not desired, ignore the message. Otherwise, run the BAD program on the disk; manually enter bad block data; or re-enter the command using a new disk. The functions of the BAD utility are described in the TRAX Support Environment System Operations Guide.

DSC – 67 OUTPUT DEVICE device: IS A DIAGNOSTIC PACK DO NOT USE IT!

The specified output disk is a diagnostic device, and cannot be used.

Mount new output disk and re-enter the command.

DSC – 68 CODE B ON device: file id file number VBN: expected x found y

The tape position was lost when reading the virtual block number specified. Some data may be lost.

Determine the extent of the error. If necessary, try the tape on another drive, or create another tape.

DSC – 69 CODE C ON device: file id file number VBN

The position of the tape was lost while reading the data file specified. Data beyond the VBN mentioned are lost.

Re-create the tape, or retry the operation on a different tape drive.

DSC – 70 CODE D ON device: file id file number expected x found y

The tape position was lost while reading the tape mentioned in the message. All of "y" and some of "x" are lost.

Retry the entire operation.

DSC – 71 FAILED TO MAP OUTPUT FILE ON device: file id file number

An inconsistency occurred when writing the specified file to the output disk. The file header did not specify the correct number of virtual blocks required to write the file and the file is lost.

Retry the operation.

DSC – 72 OUTPUT DISK device: IS TOO SMALL – nn BLOCKS NEEDED

The output disk is not large enough to accommodate the data to be transferred.

Retry the operation specifying a larger output disk.

DSC – 73 I/O ERROR C ON device:

The following message explains the error that occurred while reading the specified file.

Retry the operation.

DSC – 74 I/O ERROR H ON device:

The message that follows explains the I/O error that occurred while writing the specified file.

Retry the operation.

DSC – 75 I/O ERROR J ON device:

An I/O error (which follows) occurred when reading the tape labels on the specified device.

Retry the operation on a different tape drive.

DSC – 76 INPUT TAPE ON device: MUST BE AT BOT

The specified tape must be at beginning of tape or its load point. This message is also displayed during a /VE operation merely to indicate that the current volume is rewinding to enable the verify pass.

If /VE was not specified, check the tape and remount at load point.

DSC – 77 WRONG INPUT TAPE ON device: EXPECTING file id FOUND file id

The input tapes were specified out of sequence.

Check the tapes, re-enter in proper order after receiving mount instructions.

DSC – 78 CODE E ON device: AFTER file id file number

This is the result of a read error from tape. When trying to read an attribute block, some other block was accessed. The file following the file specified in the error message is lost.

Retry the operation.

DSC – 79 I/O ERROR K ON device:

The message that follows explains the I/O error that occurred while reading the specified file.

Retry the operation.

DSC – 80 I/O ERROR L ON device:

The message that follows explains the I/O error that occurred while reading the file header.

Retry the operation.

DSC – 81 INPUT TAPE device: RESYNCHRONIZED AT file id file number

The tape position has been recovered. Some data preceding the file specified were lost.

This is usually received in conjunction with one or more error messages, all indicating that the input tape was either read incorrectly or recorded badly. The tape should be re-created and the operation re-initiated.

DSC – 82 TAPE FILE filelabel NOT FOUND ON device:

The input tape specified does not contain the file identified as "filelabel".

Check the filelabel and the tape, re-enter when the correct tape and filelabel are specified.

DSC – 83 EXPECTED EXTENSION HEADER NOT PRESENT ON device: file id, file number

A tape read error occurred causing the specified file to be lost.

If the error message was preceded by one or more I/O warning messages, the operation should be retried. If not, the input tape is bad and should be re-generated.

DSC – 84 CODE F ON device: AFTER file id file number

This is the result of a read error from tape. When trying to read a file header some other block type was accessed. The file following the file specified in the error message is lost.

Retry the operation.

DSC – 85 I/O ERROR M ON device:

The following message explains why the specified file could not be read.

Retry the operation.

DSC – 86 INDEX FILE DATA NOT PRESENT device:

When reading the input tape specified, a file other than the index file was accessed due to a tape error or an I/O error.

Re-create the tape, or retry the same tape on a different tape drive.

DSC – 87 I/O ERROR N ON device:

The message that follows explains the I/O error that occurred while restoring the index and storage map files from the specified input tape.

Retry the operation using a different input tape drive.

DSC – 88 VOLUME SUMMARY DATA NOT PRESENT device:

Either the input tape is not a DSC tape, or incomplete data are contained.

Check the tape and re-enter the command.

DSC – 89 I/O ERROR O device: file id, file number

The message that follows explains the I/O error that occurred while writing the specified file header.

Retry the operation.

NOTE

The DSC errors identified as I/O errors are accompanied by one or more of the following error messages to explain the type of i/o error that occurred.

BAD BLOCK NUMBER

The block does not exist on the disk; an internal DSC error has occured; or the block is bad.

Retry the operation with a new disk and/or disk drive.

BAD BLOCK ON DEVICE

A device malfunctions has occurred, or a tape was used with bad data on it resulting in a block containing incorrect information.

Retry the operation.

BLOCK CHECK

A parity error occurred indicating that bad data may have been transferred.

Retry the operation.

DATA OVERRUN

The physical tape used is larger than was expected; the tape got out of position, or is in the wrong format.

Make sure the tape is the right one and retry the operation.

DEVICE NOT READY

The device is not ready or not up to speed, or a blank tape has been used as an input tape.

Retry the operation after checking that the device is online and correctly mounted.

DEVICE OFFLINE

The device is not in the system.

Check the device, the device specification in the command string, and re-enter the command.

DEVICE WRITE LOCKED

The disk drive is write locked.

Write enable the disk drive and re-enter the command.

END OF FILE DETECTED

The tape position was lost.

Retry the operation.

END OF TAPE DETECTED

The tape position was lost.

Retry the operation.

END OF VOLUME DETECTED

The tape position was lost.

Retry the operation.

FATAL HARDWARE ERROR

A hardware malfunctions has occurred.

Retry; if error repeats call DIGITAL Field Service.

INSUFFICIENT POOL SPACE

The operating system is overloaded.

Retry the operation.

PARITY ERROR ON DEVICE

A device malfunctions or tape incompatibility has occurred.

Retry the operation.

PRIVILEGE VIOLATION

A device has been mounted as FILES-11.

TRAX users: DISMOUNT the disk and retry the operation.

UNKNOWN SYSTEM ERROR

An undefinable I/O error has occurred.

Retry the operation.

The following error messages appear only in the stand-alone version of DSC used as part of the SYSGEN procedure.

Table B-1 General Error and I/O Error Message Codes

General Error Message Codes

Symbol	Meaning
Code A	Failed to read storage map header
Code B	Input data out of phase
Code C	Non-data block encountered
Code D	Input file out of phase
Code E	File attributes out of phase
Code F	File header out of phase

I/O Error Message Codes

Symbol	Meaning
A	Reading index file bit map
B	Reading index file header
C	Reading storage bit map
D	Reading boot or home block
E	Reading file header
F	Input (or output device)
G	Writing index file bit map
H	Writing storage bit map header
I	Reading input device
J	In input tape labels
K	Reading file attributes
L	Reading file header
M	Reading index file data
N	Reading summary data
O	Writing file header

utl – DEVICE/FILE IS FULL – device/filename

Description

The utility cannot create an output file on the indicated device because of insufficient space or the indicated file cannot be extended due to insufficient space.

Suggested User Action

Reenter the command using another device for output files or copy the indicated file to another device and retry the command. Optionally, delete unneeded files on the indicated device and reenter the original command line.

?utl – ERROR WITH WILDCARDS

Description

The wild card processor has returned an error to the utility during resolution of wild cards in a file specification.

Suggested User Action

Reenter the command line. If the same condition recurs, use successive invocations of the utility and non-wild carded file specifications to achieve the original desired result.

?utl – FILE NOT AVAILABLE – filename

Description

The indicated file is being accessed for exclusive use by another job.

Suggested User Action

Periodically retry the command until the file has been released.

?utl – FILE POSITION LOST – filename

Description

The utility has lost its position within a container file on magnetic tape while rewinding or backspacing. The error may have been caused by hardware failure.

Suggested User Action

Determine from the output account or summary listing file the extent of the processing that was completed prior to the occurrence of the error. Reenter the command line eliminating file specifications of files successfully processed. Use a new tape volume and/or a different tape drive. The file is input to RESORE, the utility cannot restore the data records within the indicated blocks of the original file.

?utl – I/O ERROR ENCOUNTERED ON OUTPUT FILE – filename

Description

One of the following conditions exists:

1. The device is not on line.
2. The device is not mounted.
3. The hardware has failed.
4. The volume is full.

Suggested User Action

Rectify the condition and reenter the command line.

utl – INPUT FILE IS NOT BACKUP FILE – filename

Description

The utility requires that the input file be a backup file. The user has specified a file that is not in backup format. For example, a file not in backup format is specified as input to ARCHIVE/RESTORE.

Suggested User Action

Reenter the command line with the correct file specification.

utl – NO SUCH FILE

Description

No files in the UFD correspond to the wild cards of a file specification.

Suggested User Action

Obtain a listing of the files in the desired UFD. Reenter the command with the desired file specification.

?utl – PRIVILEGE VIOLATION – filename

Description

The user does not have the privileges necessary to access the indicated file.

Suggested User Action

Have the owner of the file change its privilege specification.

utl – READ ERROR, INTEGRITY CHECK TABLE AND REWRITE DATA MAY HAVE BEEN LOST

Description

The utility has encountered an error while attempting to read internal data maintained in a backup file or container file.

Suggested User Action

Retry the command. If the same error occurs, check summary listing file created at the time the backup or container file was created. Determine from this file which file or files cannot be completely restored.

utl – READ ERROR ON FILE ATTRIBUTES – filename

Description

The volume is corrupted or the user does not have the necessary privileges to access the indicated file.

utl – READ ERROR ON FILE PROLOGUE – filename

Description

The utility is unable to read the prologue (internal RMS-11 information within a file) of the specified file. The file is bypassed and processing continues.

Suggested User Action

Reenter the command specifying the subject file only. If the same error occurs, the indicated file cannot be properly accessed on the device. Use the RESTORE utility to retrieve a new copy of the file.

utl – READ ERROR OR INCONSISTENT DATA. MAY HAVE LOST FILES.

Description

The utility has encountered an error while reading a backup or container file.

Suggested User Action

Retry the command. If the same error occurs, check summary listing file created at the time the backup or container file was created. Determine from this file which file or files may have been lost.

?utl – REWIND OR SPACE ERROR ON FILE – filename

Description

The utility has encountered an error while rewinding or backspacing on magnetic tape.

Suggested User Action

Retry the command. If the condition occurs again, mount the volume on another drive and retry the command.

?utl – SELECT ERROR – dev

Description

One of the following conditions exists:

1. The device is not on-line.
2. The device is not mounted.
3. The hardware has failed.

Suggested User Action

Rectify the condition and reenter the command line.

utl – UNABLE TO RESTORE SPECIAL ATTRIBUTES – filename

Description

The utility was unable to restore the file with one or more of its original date attributes (e.g., creation date, revision date) or its original protection specification.

Suggested User Action

Use the DISPLAY utility to determine which attributes of the file were not restored.

?utl – WRITE ERROR ON ATTRIBUTES OF FILE – filename

Description

The volume is corrupted or the user does not have the necessary privileges to write the file.

Suggested User Action

Verify access to file.

?utl – WRITE ERROR ON CREATE OF OUTPUT FILE – filename

Description

One of the following conditions exists:

1. The device is not on line.
2. The device is not mounted.
3. The hardware has failed.
4. The volume is full.

Suggested User Action

Rectify the coddition and reenter the command line.

?utl – WRITE ERROR ON INTEGRITY CHECK TABLE ON OUTPUT FILE – filename

Description

The utility is unable to write internal data integrity checking tables in the output backup file.

Suggested User Action

If the output medium is magnetic tape, use a different tape volume and retry the command. If the output medium is disk, rename the output file so that the utility will not attempt to use the space and retry the command.

B.5 COPY

These are the error messages created by the COPY command.

COP – ALLOCATION FAILURE – NO CONTIGUOUS SPACE

Description: Contiguous space available on the output volume is insufficient for the file being copied.

Suggested User Response: Delete all files that are no longer required on the output volume, and reenter the command line.

COP – ALLOCATION FAILURE ON OUTPUT FILE

or

COP – ALLOCATION FAILURE – NO SPACE AVAILABLE

Description: Space available on the output volume is insufficient for the file being copied.

Suggested User Response: Delete all files that are no longer required on the output volume, and reenter the command line. Also, use the archive command.

COP – BAD USE OF WILD CARDS IN DESTINATION FILE NAME

Description: A wildcard * was specified for an output filename where use of a wildcard is explicitly disallowed.

Suggested User Response: Reenter the command line with the proper output file explicitly specified.

COP – CANNOT FIND DIRECTORY FILE

Description: UFD specified does not exist on this volume.

Suggested User Response: Reenter the command line, specifying the correct UFD or the correct volume.

COP – CANNOT FIND FILES(S)

Description: The files(s) specified in the command were not found in the designated directory.

Suggested User Response: Check the file specifier and reenter the command line.

COP – CLOSE FAILURE ON INPUT FILE

or

COP – CLOSE FAILURE ON OUTPUT FILE

Description: The input or output file could not be properly closed. The file is locked to indicate possible corruption.

Suggested User Response: Reenter the command line. If the error recurs, run a validity check of the file structure using the verify utility (VFY) on the volume in question to determine if it is corrupted. The functions of the Verify utility are described in the TRAX System Manager's Guide.

COP – DIRECTORY WRITE PROTECTED

Description: COP could not remove an entry from a directory because the device was write-protected, or because of privilege violation.

Suggested User Response: Enable the unit for write operations or have the owner of the directory change its protection.

COP – FAILED TO ENTER NEW FILE NAME

Description: You have specified a file that already exists in the directory file, or do not have the necessary privileges to make entries in the specified directory file.

Suggested User Response: Reenter the command line, ensuring that the filename and UFD are specified correctly, or request COP under the correct UIC and reenter the command line.

COP – FAILED TO WRITE ATTRIBUTES

Description: Volume is corrupted or you do not have the necessary privileges to write the file attributes.

Suggested User Response: Ensure that COP is running under the correct UIC. If the UIC is correct, then run the validity check of the file structure verification utility (VFY) against the volume in question to determine where and to what extent the volume is corrupted. The functions of the Verify utility are described in the TRAX System Manager's Guide.

COP – ILLEGAL "*" COPY TO SAME DEVICE AND DIRECTORY

Description: You attempted to copy all versions of a file into the same directory that is being scanned for input files. This results in an infinite number of copies of the same file and is not allowed.

Suggested User Response: Reenter the command line, renaming the files or copying them into a different directory.

COP – I/O ERROR ON INPUT FILE

or

COP – I/O ERROR ON OUTPUT FILE

Description: One of the following conditions may exist:

- The device is not on-line.
- The device is not mounted.
- The hardware has failed.
- The volume is full (output only).
- Input file is corrupted.

Suggested User Response: Determine which condition caused the message and correct that condition. Reenter the command line.

COP – NO SUCH FILE(S)

Description: The file(s) specified in the command were not found in the designated directory.

Suggested User Response: Check the file specifier and reenter the command line.

COP – OPEN FAILURE ON INPUT FILE

or

COP – OPEN FAILURE ON OUTPUT FILE

Description: The specified file could not be opened. One of the following conditions may exist:

- The file is protected against access.
- A problem on the physical device (e.g., device cycled down).
- The volume is not mounted.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.

Suggested User Response: Determine which condition caused the message and correct that condition. Reenter the command line.

B.6 CREATE

These are the error messages created by the CREATE command.

NOTE

A question mark [?] preceeding the dfn utility message indicates a fatal error. A question mark in brackets [?] indicates that the error may be fatal or diagnostic. If no question mark preceeds the error message it is a diagnostic.

?dfn – DEVICE OFF LINE - device

Description

The indicated device exists on the system but the attempt to access it has been prohibited for one of the following reasons.

1. The device is not ready.
2. No volume is mounted on the device.
3. The device is currently reserved by another job.
4. The device requires privileges for ownership and the user does not have privilege.
5. The device has been disabled.

Suggested User Action

Determine the nature of the problem and take corrective action.

?dfn – DEVICE WRITE PROTECTED – device

Description

The utility cannot access the indicated device for write operations.

Suggested User Action

Check the hardware condition of the indicated device. Write enable the unit.

?dfn – DEVICE/FILE IS FULL – device/filename

Description

The utility cannot create an output file on the indicated device because of insufficient space or the indicated file cannot be extended due to insufficient space.

Suggested User Action

Reenter the command using another device for output files or copy the indicated file to another device and retry the command. Optionally, delete unneeded files on the indicated device and reenter the original command line.

?dfn – DIRECTORY NOT FOUND – filename

Description

The directory does not exist on the specified device.

Suggested User Action

Reenter the command with the correct directory specification.

?dfn – FILE ALREADY EXISTS – filename

Description

The utility has attempted to create an output file that already exists in the output account.

Suggested User Action

Reenter the command line using a new or corrected filename or delete the existing file and re-enter the original command line.

?dfn – ILLEGAL DEVICE – device

Description

The indicated device does not exist.

Suggested User Action

Reenter the command line with a corrected device specification.

?dfn – ILLOGICAL DEVICE – device

Description

The indicated device is not permitted in the context of the command line. For example, the user cannot CREATE an indexed file on magnetic tape.

Suggested User Action

Reenter the command line with an appropriate device specification.

?dfn – PRIVILEGE VIOLATION – filename

Description

The user does not have the privileges necessary to create the indicated file.

Suggested User Action

Have the owner of the file change its privilege specification.

?dfn – WRITE ERROR ON CREATE OF OUTPUT FILE – filename

Description

One of the following conditions exists:

1. The device is not on line.
2. The device is not mounted.
3. The hardware has failed.
4. The volume is full.

Suggested User Action

Rectify the condition and reenter the command line.

The following messages pertain to the **"CREATE/DIRECTORY" Command.**

UFD – DIRECTORY ALREADY EXISTS

The requested UFD already existed on the volume.

UFD – FAILED TO CREATE DIRECTORY

No space existed on the volume, or an I/O error occurred.

UFD – NOT FILES-11 DEVICE

The device on which the UFD was to be created was not a Files-11 device, and therefore could not support UFD's.

UFD – WRITE ATTRIBUTES FAILURE

An error was encountered while writing the attributes of either the MFD or the newly created UFD.

UFD – WRONG VOLUME

The volume label and the label specified in the command did not match.

UFD – VOLUME NOT MOUNTED

The volume on which a UFD is to be created must be mounted before accessing the files-11 structure.

B.7 DCL

The error messages described in this section are command independent. The system prefixes the error message with a unique 3 letter code derived from the command name. For example:

MES – YOU DO NOT HAVE THE PRIVILEGE TO ISSUE THIS COMMAND

the 3 letter code "MES" is derived from the MESSAGE command. In this section the command names are substituted with "XXX" to signify that the operating task mnemonic is inserted in this position.

XXX – ILLEGAL FUNCTION

Description: the command line contains an illegal command name.

Use the help command to get a list of valid commands.

XXX – SYNTAX ERROR

Description: The command line has a syntax error.

Suggested User Response: Consult Chapter 9 for the correct syntax. Reenter the command line.

XXX – FUNCTION NOT UNIQUE

Description: You did not enter enough characters to uniquely identify the command.

Suggested User Response: Consult Chapter 9 for the correct spelling of the command. Reenter the command line.

XXX – ILLEGAL QUALIFIER

Description: The command line contains an illegal qualifier for the command.

Suggested User Response: Consult Chapter 9 for the correct task qualifiers. Reenter the command line.

XXX – QUALIFIER NOT UNIQUE

Description: You did not enter enough characters to uniquely identify the qualifier in the current context.

Suggested User Response: Consult Chapter 9 for the correct format. Reenter the command line.

XXX – REQUIRED PARAMETER NOT SPECIFIED

Description: a required parameter has been omitted from the command line.

Suggested User Response: Consult Chapter 9 for the correct format. Reenter the command line.

XXX – INVALID PROTECTION CODE SPECIFIED

Description: a protection code other than R, W, E or D was specified or the protection codes where not specified in the order RWED.

Suggested User Response: check the command line. Reenter the command line correctly.

XXX – FILE SPECIFICATION EITHER INVALID OR NOT SPECIFIED

Description: the command line contains an invalid file specification or has been omitted.

Suggested User Response: check the command line. Reenter the command line correctly.

XXX – PRIMARY KEY NOT SPECIFIED

Description: The number parameter of the key qualifier is missing. It should be 1.

Suggested User Response: Check the command line. Reenter the command line correctly.

XXX – CONTRADICTORY QUALIFIER IN KEY SPECIFICATION

Description: a contradictory pair of qualifiers was specified in the key specification. For example:

UPDATE/NOUPDATE – OR – DUPLICATE/NODUPLICATE

Suggested User Response: Consult Chapter 9 for the correct format. Reenter the command line.

XXX – INVALID KEY QUALIFIER VALUE

Description: A negative number or 0 was specified as the key qualifier value.

Suggested User Response: Consult Chapter 9. Reenter the command line.

XXX – REQUIRED VALUE NOT SPECIFIED FOR POSITION SIZE OR NUMBER

Description: The numeric value for either the number, position or size qualifiers have not been specified.

Suggested User Response: Reenter the command line.

XXX – TASK ACTIVE

Description: You attempted to executive a command twice simultaneously on the same terminal.

Suggested User Response: Wait till the first invocation is completed. Reenter the command line.

XXX – WILD CARDS NOT PERMITTED

Description: Filename, type, or the version number of the file must be expressed explicitly, the wildcard default "*" cannot be used.

Suggested User Response: Reenter the command line.

XXX – ZERO VALUE NOT VALID FOR KEY SIZE OR NUMBER

Description: the key, size or number qualifiers must be a positive number.

Suggested User Response: Consult the command description in Chapter 9. Reenter the command line.

XXX – CONTRADICTORY QUALIFIER

Description: The command line contains a pair of qualifiers that are contradictory. For example:

UPDATE/NOUPDATE

XXX – INVALID FILE SPECIFICATION QUALIFIER

Description: the command line contains an invalid file specification qualifier.

Suggested User Response: check the command line. Consult the command description in Chapter 9. Reenter the command line.

XXX – COMMAND LINE INCOMPLETE

Description: a necessary parameter was omitted from the command line.

Suggested User Response: Consult the command description in Chapter 9. Reenter the command line.

XXX – LIBRARY INVALID ON LAST INPUT FILE

Description: the library must be specified before all other input files.

Suggested User Response: Consult the command description in Chapter 9.

XXX – INVALID COMMAND FUNCTION

Description: the command option is invalid. For example:

SET TABLE

Suggested User Response: Consult the command description in Chapter 9. Reenter the command line.

XXX – QUEUE MARKED FOR DELETE

Description: the command line tried to access a queue that is marked for deletion.

Suggested User Response: try another queue.

XXX – PROCESSOR MARKED FOR REMOVAL

Description: the command line tried to access a processor marked for removal.

Suggested User Response: try another processor.

XXX – QUEUE DIRECTORY FULL

Description: The maximum number of queues of a given kind have already been created.

Suggested User Response: check queue directory, delete non-essential queues as required.

XXX – PROCESSOR DIRECTORY FULL

Description: The maximum number of processors of a given kind have already been created.

Suggested User Response: check processor directory, delete non-essential processors as required.

XXX – REDUNDANT OPERATION

Description: Self-explanatory. For example, stopping a queue which is already stopped.

Suggested User Response: check command line. Reenter the command line.

XXX – DEVICE DOES NOT EXIST

Desciption: the device specified in the command line does not exist. Perhaps the unit number has been omitted.

Suggested User Response: check the device address with a show device command. Reenter the command line.

XXX – COMMAND PROCESSING TASK NOT IN SYSTEM

Description: The queue manager has not been initiated and therefore no queue management commands can be processed.

Suggested User Response: Start the queue manager.

XXX – CONFLICTING QUALIFIER

Description: A qualifier has been specified when some other attribute of the command makes it meaningless.

Suggested User Response:

XXX – RESERVED QUEUE NAME

Description: You have attempted to create a queue whose name is reserved by the system.

Suggested User Response: Choose a different queue name.

XXX – ENTRY is not a job entry

Description: You have tried to access a queue entry which is not associated with a batch or print job.

Suggested User Response: Invoke the show queue command to ascertain the correct job entry number.

B.8 DISMOUNT

These are the error messages created by the DISMOUNT command.

DMO – ALREADY MARKED FOR DISMOUNT

The device-unit had been requested to be dismounted and was in the process of waiting for all accesses to the volume to complete.

DMO – CHECKPOINT FILE STILL ACTIVE

The command attempted to dismount a volume that contained an active checkpoint file. The volume cannot be dismounted until the checkpoint file has been discontinued. In order to dismount your system disk please run the shutdown utility.

DMO – HOME BLOCK CHECKSUM ERROR

The checksum in the home block and the calculated checksum did not agree. This message is usually caused by an I/O error.

DMO – HOME BLOCK I/O ERROR

An error was detected in updating the volume file sequence number in the volume home block.

DMO – NO VOLUME LIST

The command specified a magnetic tape drive for which a mounted volume list does not exist.

DMO – NOT MOUNTABLE DEVICE

The specified device was not a mountable device and therefore could not be dismounted.

DMO – NOT MOUNTED

The specified device was not mounted.

DMO – WRONG VOLUME

The volume label and the label specified in the command did not match.

B.9 INITIALIZE

These are the error messages created by the INITIALIZE command.

INI – BAD BLOCK FILE CORRUPT - DATA IGNORED

Although automatic bad block recognition was selected, the bad block data on the disk was not in the correct format, and was therefore ignored.

INI – BAD BLOCK FILE FULL

The disk had more than 102 bad regions on it.

INI – BAD BLOCK HEADER I/O ERROR

An error was detected in writing out the bad-block file header.

INI – BLOCK(S) EXCEED VOLUME LIMIT

The specified block (or blocks) exceeded the physical size of the volume.

INI – BOOT BLOCK WRITE ERROR

An error was detected in writing out the volume boot block.

INI – CHECKING DDnn

Not an error message. An automatic bad-block specification was proceeding, using the bad-block file provided by the Bad Block Locator utility program or, on an RK07, the factory-written file from the last track of the disk.

INI – CHECKPOINT FILE HEADER I/O ERROR

An error was detected in writing out the checkpoint file header.

INI – DATA ERROR

The command specified a bad-block number or contiguous region that was too large.

INI – DISK IS ALIGNMENT CARTRIDGE

THE LAST TRACK ON AN RK07 identified the volume as an alignment cartridge, which cannot be initialized as a Files-11 volume. An alignment cartridge is specifically formatted for aligning disk read/write heads.

INI – DUPLICATE BLOCK(S) FOUND

A block that had been defined as bad was being defined as bad a second time.

INI – FAILED TO READ BAD BLOCK FILE

The command was unable to read the bad block information from the last track of an RK07 disk.

INI – HOME BLOCK ALLOCATE WRITE ERROR

In overwriting a bad-home-block area, a write error occurred.

INI – HOME BLOCK WRITE ERROR

An error was detected in writing out the volume home block.

INI – INDEX FILE BIT MAP I/O ERROR

An error was detected in writing out the index-file bit map.

INI – INDEX FILE HEADER I/O ERROR

An error was detected in writing out the index-file header.

INI – MFD FILE HEADER I/O ERROR

An error was detected in writing out the Master File Directory (MFD) file header.

INI – MFD WRITE ERROR

An error was detected in writing out a block in the Master File Directory (MFD).

INI – NO BAD BLOCK DATA FOUND

Although automatic bad-block specification was selected, no bad-block file was found on the volume.

INI – NOT FILES-11 DEVICE

The system does not support files-11 on the specified device.

INI – NULL FILE HEADER I/O ERROR

An error was detected in writing out null-file headers to the index file.

INI – STORAGE BITMAP FILE HEADER I/O ERROR

An error was detected in writing out the storage allocation file header.

INI – VOLUME MOUNTED

An attempt was made to initialize a mounted volume. Mounted volumes can not be initialized.

INI – VOLUME NOT READY

The command specified a volume that was not ready (not up to speed).

INI – VOLUME WRITE LOCKED

The command specified a volume that was write-locked and therefore could not be initialized as a Files-11 device.

INI – WARNING BLOCK 0 IS BAD

Block 0 of the specified volume, the boot block, was bad. A bootable image can therefore not be placed on this volume.

B.10 LIBRARIAN

These are the error messages created by the LIBRARIAN command.

LBR – BAD LIBRARY HEADER

Description: Either the file is not a library file or the file is corrupted.

Suggested User Response:

- If the file is not a library file, reenter the command line with a proper library file specified.
- If the file is a proper library file, the user should run the file structure verification utility (VFY) against the volume to determine if it is corrupted. The functions of the Verify utility are described in the TRAX System Manager's Guide.
- If the volume is corrupted, it must be reconstructed before it can be used.

LBR – DUPLICATE ENTRY POINT NAME "name" IN filename

Description: An attempt has been made to insert a module into a library file when both contain an identically-named entry point.

Suggested User Response: Determine if the specified input file is the correct file. If not, reenter the command line, specifying the correct input file. If the input file is the correct file, the user may delete the duplicate entry point from the library and rerun.

LBR – DUPLICATE MODULE NAME "name" IN filename

Description: An attempt has been made to insert (without replacement) a module into a library that already contains a module with the specified name.

Suggested User Response: Determine if the specified input file is the correct file. If the input file is correct, decide whether to delete the duplicate module from the library file and insert the new one, or replace the duplicate module by rerunning LBR with the /RP switch appended to the input file specifier.

LBR – EPT OR MNT EXCEEDED IN filename

Description: The EPT or MNT table limit has been reached during the execution of an Insert or Replace command.

Suggested User Response: Copy the library, increasing the table space via the COMPRESS command. Reenter the command line.

LBR – EPT OR MNT SPACE EXCEEDED IN COMPRESS

Description: An EPT or MNT table size was specified for the output library file that is not large enough to contain the EPT or MNT entries used in the input library file.

Suggested User Response: Reenter the command line with a larger EPT or MNT table size specified.

LBR – ERROR IN LIBRARY TABLES, FILE filename

Description: The library file is corrupted or is not a library file.

Suggested User Response: If the file is corrupted, no recovery is possible; the file must be reconstructed. If the file is not a library file, reenter the command line with the correct library file specified.

LBR – FATAL COMPRESS ERROR

Description: The input library file is corrupted or is not a library file.

Suggested User Response: No recovery is possible. The file in question must be reconstructed.

LBR – INVALID EPT AND/OR MNT SPECIFICATION

Description: An EPT or MNT value greater than 4096(10) was entered in a CREATE or SQUEEZE function.

Suggested User Response: Reenter the command line with the correct value specified.

LBR – INVALID FORMAT, INPUT FILE filename

Description: The format of the specified input file is not the standard format for a macro source or object file, or the input file is corrupted.

Suggested User Response: Reenter the command line with the correct input file specified.

LBR – NO ENTRY POINT NAMED "name"

Description: The entry point to be deleted is not in the specified library file.

Suggested User Response: Determine if the entry point is misspelled or if the wrong library file is specified. Reenter the command line with the entry point correctly specified.

LBR – NO MODULE NAMED "module"

Description: The module to be deleted is not in the specified library file.

Suggested User Response: Determine if the module name is misspelled or if the wrong library file is specified. Reenter the command line with the module name correctly specified.

LBR – OPEN FAILURE ON FILE filename

Description: The file system, while attempting to open a file, has detected an error. One of the following conditions may exist:

- The user directory area is protected against an open.
- A problem exists on the physical device (e.g., device cycled down).
- The volume is not mounted.
- The specified file directory does not exist.
- The file does not exist as specified.
- Insufficient contiguous space to allocate the library file (compress and create only).
- Insufficient dynamic memory in Executive.

Suggested User Response: Determine which of the above conditions caused the message and correct that condition. Reenter that command line.

LBR – OUTPUT ERROR ON filename

Description: A write error has occurred on the output file. One of the following conditions may exist:

- The volume is full.
- The device is write-protected.
- The hardware has failed.

Suggested User Response: If the volume is full, delete all unnecessary files and rerun LBR. If the device is write-protected, write-enable the device, and reenter the command line. If the hardware has failed, swap devices and reenter the command line or wait until the device is repaired and rerun LBR.

B.11 LINK

The functions of the LINK command are described in the TRAX Linker Reference Manual. The TRAX Linker produces diagnostic and fatal error messages. Error messages are printed in the following forms:

TKB – *DIAG*-error-message

or

TKB – *FATAL*-error-message

Some errors are correctable when command input is from a terminal. In such a case, a diagnostic error message can be printed, the error corrected, and the task building sequence continued. If the same error is detected in an indirect file by the TRAX Linker, a correction cannot be made, and the task linkage operation is aborted.

Some diagnostic error messages merely advise the user of an unusual condition. If the user considers the condition normal to his task, he can run the task image.

The following section tabulates the error messages produced by the Task Builder. Most of the messages are self-explanatory. In some cases, the line in which the error occurred is printed.

A Software Performance Report (SPR) should be submitted to DIGITAL in cases where the explanation accompanying a message refers to a system error.

ALLOCATION FAILURE ON FILE file-name

The TRAX Linker could not acquire sufficient disk space to store the task image file, or did not have write-access to the UFD or volume that was to contain the file.

LOOKUP FAILURE ON FILE filename
invalid-line

The invalid-line printed contains a filename that cannot be located in the directory.

LOOKUP FAILURE ON SYSTEM LIBRARY FILE

The TRAX Linker cannot find the system Library (SYO:[1,1] SYSLIB.OLB) file to resolve undefined symbols.

LOOKUP FAILURE RESIDENT LIBRARY FILE
invalid-line

No symbol table or task image file can be found for the shared region.

MODULE module-name NOT IN LIBRARY

The TRAX Linker could not find the module named on the LB switch in the library.

SEGMENT seg-name HAS ADDR OVERFLOW: ALLOCATION DELETED

Within a segment, the program has attempted to allocate more than 32K. A map file is produced, but no task image file is produced.

TASK HAS ILLEGAL MEMORY LIMITS

An attempt has been made to build a task whose size exceeds the partition boundary. If a task image file was produced, it should be deleted.

TASK IMAGE FILE filename IS NON-CONTIGUOUS

Insufficient contiguous disk space was available to contain the task image. A non-contiguous file was created. After deleting unnecessary files, the /CO switch in PIP should be used to create a contiguous copy.

B.12 LOGIN

The following are the error messages created by the LOGIN command.

LOG – ACCOUNT FILE OPEN FAILURE

The account file was open for another user; or the disk containing the account file was not mounted. Retry command.

LOG – INVALID ACCOUNT

The name or UIC specified in the command is not stored in the account file; or the password specified does not match the name or UIC given.

LOG – LOGINS ARE DISABLED

The system was in the process of shutting down; or the command SET NOLOGIN has been issued. A user cannot log onto a terminal at these times.

LOG – MESSAGE FILE ERROR nnn.

The system could not open the file [1,2] LOGIN.TXT for a reason indicated by the FCS code nnn. See Section x.xx for a definition of the FCS code.

LOG – OTHER USER LOGGED ON

The issuing terminal was currently logged by another user. Only one user at a time can be logged onto a terminal.

LOG – TERMINAL ALLOCATED TO OTHER USER

The issuing terminal has been allocated to another user. A user cannot log onto a terminal allocated to someone else.

The system was unable to allocate a system file from the specified block because of intermediate bad blocks or end of volume.

B.13 MERGE

These are the error messages created by the MERGE command.

NOTE

A question mark "?" preceeding the cnv error message indicates a fatal error. A question mark in brackets [?] indicates that the error may be fatal or diagnostic. If no question mark preceeds the error message it indicates a diagnostic error.

?cnv – DEVICE OFF LINE - device

Description

The indicated device exists on the system but the attempt to access it has been prohibited for one of the following reasons.

1. The device is not ready.
2. No volume is mounted on the device.
3. The device is currently reserved by another job.
4. The device requires privileges for ownership and the user does not have privilege.
5. The device has been disabled.

Suggested User Action

Determine the nature of the problem and take corrective action.

?cnv – DEVICE WRITE PROTECTED - device

Description

The utility cannot access the indicated device for write operations.

Suggested User Action

Check the hardware condition of the indicated device. Write enable the unit.

cnv – DEVICE/FILE IS FULL - device/filename

Description

The utility cannot create an output file on the indicated device because of insufficient space or the indicated file cannot be extended due to insufficient space.

Suggested User Action

Reenter the command using another device for output files or copy the indicated file to another device and retry the command. Optionally, delete unneeded files on the indicated device and reenter the original command line.

?cnv – DIRECTORY NOT FOUND - filename

Description

The directory does not exist on the specified device.

Suggested User Action

Reenter the command with the correct directory specification.

cnv – DUP RCD=string

Description

The utility could not write a record into an indexed file because duplicate key values for one or more keys in the record were not permitted. Writing the record would cause duplication. The displayed string represents the first 72 characters of the record that could not be written.

Suggested User Action

None.

cnv – number DUPLICATE RECORDS NOT WRITTEN

Description

The utility could not write the indicated number of records into an indexed file because duplicate key values for one or more keys were not permitted.

Suggested User Action

None.

?cnv – FILE NOT AVAILABLE - filename

Description

The indicated file is being accessed for exclusive use by another job.

Suggested User Action

Periodically retry the command until the file has been released.

[?] cnv – FILE NOT FOUND - filename

Description

The indicated file was not found in the designated UFD and device.

Suggested User Action

Verify the file specification and reenter the command line.

?cnv – FILE READ ERROR

Description

The utility has encountered a hardware read error on an input or output device.

Suggested User Action

If not at TRAX Support Environment prompt level, use CTRL/Z to terminate access to utility. Check input and output devices for hardware problems.

?cnv – ILLEGAL DEVICE - device

Description

The indicated device does not exist.

Suggested User Action

Reenter the command line with a corrected device specification.

?cnv – INPUT AND OUTPUT RECORD FORMATS DO NOT CORRESPOND

Description

The user is attempting to write records from one file to another. However, the input file records are variable and the output file records are fixed.

?cnv – INPUT AND OUTPUT RECORD SIZES DO NOT CORRESPOND

Description

The user is attempting to write records from one file to another. However, one of the following conditions exists:

1. Both files have fixed format records but the fixed size differs.
2. Both files have variable format records but the maximum size of the input file is greater than the maximum size of the output file.

Suggested User Action

Redefine the output file and retry the command.

?cnv – MAXIMUM RECORD EXCEED - filename

Description

No more records can be written into the indicated relative file because of the file's maximum number of records attribute.

Suggested User Action

Create a new relative file through a MACRO-11 program. Specify an appropriate MRN (maximum record number) attribute. Rerun the utility.

?cnv – NO SUCH KEY FOR FILE - value

Description

The specified key of reference value represents a non-existent key in an indexed file.

Suggested User Action

Reenter the command with a correct key of reference value.

?cnv – PRIVILEGE VIOLATION - filename

Description

The user does not have the privileges necessary to access the indicated file.

Suggested User Action

Have the owner of the file change its privilege specification.

?cnv – RECORD TOO BIG - filename

Description

A record from the indicated input file exceeds the maximum record size attribute of the output file.

Suggested User Action

Use the DEFINE utility to create a new file with an appropriate maximum record size.

B.14 MOUNT

These are the error messages created by the MOUNT command.

MOU – ACP NOT IN SYSTEM

The task specified as ACP or default ACP was not installed in the system.

MOU – ALREADY MOUNTED

The specified device-unit was already mounted.

MOU – DEVICE ATTACHED [-dev:]

The device-unit specified in the command was attached by a task and could not be mounted. For attempts to mount one or more magnetic tapes, the message includes a specific device-unit.

MOU – DEVICE OFFLINE [-dev:]

The device specified in the command, although generated into the system, was not physically present in the host configuration. If the offline device is a magnetic tape drive, the message includes the device-unit.

MOU – FILE HEADER READ ERROR

Mount could not read either the index file header or the storage allocation file.

MOU – HOME BLOCK READ ERROR

An I/O error was detected in trying to read the home block. This message usually indicates that the volume is not ready. Wait until it is ready and reissue the command.

MOU – MOUNT ERROR FROM ACP xxx.

The ACP detected an error while trying to mount the volume set.

MOU – NOT MOUNTABLE DEVICE

The specified device was not supported as a Files-11 device (including ANSI magnetic tape) or a network device.

MOU – OTHER VOLUME MOUNTED [-dev:]

An attempt was made to mount a volume on a device that already had a mounted volume. The message specifies the device-unit if it is a tape drive.

MOU – STORAGE BIT MAP FILE READ ERROR

An I/O error was encountered while reading the storage allocation file.

MOU – WRONG VOLUME

The volume label and the label specified in the command did not match.

MOU – VOLUME STRUCTURE NOT SUPPORTED

TRAX did not support the files-11 structure level of the volume being mounted.

B.15 RENAME

These error messages are created by the RENAME command.

REN – CANNOT FIND DIRECTORY FILE

Description: UFD specified does not exist on this volume.

Suggested User Response: Reenter the command line, specifying the correct UFD or the correct volume.

REN – CANNOT RENAME FROM ONE DEVICE TO ANOTHER

Description: You attempted to rename a file across devices.

Suggested User Response: Reenter the command line, renaming the file on the input volume; then, enter another command to transfer the file to the intended volume.

REN – DIRECTORY WRITE PROTECTED

Description: REN could not remove an entry from a directory because the device was write-protected, or because of privilege violation.

Suggested User Response: Enable the unit for write operations or have the owner of the directory change its protection.

B.16 SET

These are the error messages created by the SET command.

SET – DEVICE NOT VARIABLE SPEED MULTIPLEXER

An attempt was made to set the baud rate for a terminal that was not attached to a DZ11 multiplexer.

SET – DEVICE NOT TERMINAL

An attempt was made to set terminal characteristics for a nonterminal device.

SET – INVALID SPEED

The multiplexer line specified does not support the requested speed; or the command specified unequal receive and transmit speeds for a DZ11. The DZ11 does not support split speeds.

SET – LINE NOT DZ11

The command attempted to set to remote a line that was not attached to a DZ11 multiplexer.

B.17 SORT

The functions of the SORT command are described in the TRAX Sort Reference Manual. The error messages generated by the SORT command are displayed in two formats:

The first format is:

SORT ERROR – CODE nn

The following is a list of the SORT error codes and a brief explanation of their meaning:

SORT ERROR – CODE 00

Description: No errors.

SORT ERROR – CODE 01

Description: Device input error.

SORT ERROR – CODE 02

Description: Device output error.

SORT ERROR – CODE 03

Description: OPEN(IN) failure.

SORT ERROR – CODE 04

Description: OPEN(OUT) failure.

SORT ERROR – CODE 05

Description: Size of current record is greater than maximum size.

SORT ERROR – CODE 06

Description: Not enough work area.

SORT ERROR – CODE 07

Description: RETRN was called after it had exited with a negative error code (end of sort).

SORT ERROR – CODE 10

Description: SORT routine called out of order. (The order of the calls should be RSORT, RELES, MERGE, RETRN, ENDS).

SORT ERROR – CODE 11

Description: Sort already in progress. (To do a second sort, ENDS must be called to clean up the first sort.)

SORT ERROR – CODE 12

Description: Key size is not positive, Sorts detected a zero or negative key size in its calling parameter.

SORT ERROR – CODE 13

Description: Record size is not positive.

SORT ERROR – CODE 14

Description: Key address is not even. (The keys must start at an even address because SORT uses word moves).

SORT ERROR – CODE 15

Description: Record address is not even.

SORT ERROR – CODE 16

Description: Scratch records will be too large (the size of the keys plus the size of the largest record must be less than 37776 octal).

SORT ERROR – CODE 17

Description: Too few scratch files are given (a minimum of 3 scratch files must be specified).

SORT ERROR – CODE 20

Description: Too many scratch files are given (a maximum of 10 scratch files may be specified).

SORT ERROR – CODE 21

Description: End-of-string record was detected where none was expected.

SORT ERROR – CODE 22

Description: Unexpected end-of-file.

SORT ERROR – CODE 23

Description: SORT found a record larger than expected.

SORT ERROR – CODE 24

Description: Record length is not standard for SORTT, SORTA, SORTI.

The second format is:

SRT – control-phase:?message [-RMS-status-code]

where:

control-phase	is the SORT phase in control at the time the error occurred. These values are: C - command decoder M - merge P - presort
message	is a one-line brief explanation of what happened.
RMS-status-code	is a decimal status code returned by RMS for additional information on file errors only. If RMS is not impacted by the SORT error, this status code does not appear. Status codes likely to be seen are listed with their meanings in Section 4.7.

SRT – C:?SORT COMMAND ERROR

a. Too many input files (more than two, including specification file) or output files (more than one)
b. General syntax error
c. Too many switches
d. Erroneous switches on the specification file
e. An undefined switch

SRT – C:?IMPROPER SWITCH: /FI

a. Less than three or greater than eight scratch files.
b. Invalid terminator

NOTE

Valid terminators are period, comma, slash, equal sign and <CR>, "INVALID TERMINATOR" means that some other character was used as a terminator or that SORT expected to find a terminator where none existed.

SRT – C:?IMPROPER SWITCH: /KE

a. Invalid letter or value
b. Start location or size is 0
c. No period (.) between start location and size
d. Illegal size for data mode
e. Invalid terminator (See NOTE above)

SRT – C:?TOO MANY KEYS

Buffer space overflowed

NOTE

SORT reserves a buffer area for storage of a table based on the input specifications in order to control the processing of each record. This space should be ample for all situations to make this error unlikely.

SRT – C:?NO KEYS SPECIFIED

There are no key switches in the command string and no specification file has been declared.

SRT – C:?KEY AFTER LAST BYTE OF RECORD

The end of an input record key field goes past the stated record size (switch or specification).

SRT – C:?NO /FO SWITCH

You omitted the /FORMAT switch on the input file.

SRT – C:?IMPROPER SWITCH: /FO

You have not specified a valid format type.

SRT – C:?IMPROPER SWITCH: /PR

You specified an invalid sort process.

SRT – C:?INVALID CHARACTER [RMS-Status-Code]

a. Column 6 is not H, I, O, F and record is not ALTSEQ.
b. Process is not SORTR, SORTT, SORTA, SORTI, or blank.
c. Collating sequence is not blank, E, or X.
d. Data type is not B, C, D, F, I, J, K, P, Z.
e. Key type is not D, F, N, O.
f. Logical entry is not A, O, blank, or *.

SRT – C:?ILLEGAL FIELD [RMS-Status-Code]

a. A numeric field in specification contains other than decimal digits or blanks.
b. No key size is given in Header specification.
c. No output size is given in Header Specification if type of SORT is SORTR or SORTT.
d. ALTSEQ is misspelled.
e. ALTSEQ entries do not represent 7-bit octal values.
f. Last location is less than first location in record field identification.
g. Size is invalid for data mode.
h. Sizes of Factors 1 and 2 in Record Specification do not match.
i. Compare relation is undefined.
j. Forced field is other than type C or more than one position.

SRT – C:?ILLEGAL CONSTANT [RMS-Status-Code]

a. Constant given in Factor 2 is greater than 20 characters.
b. Mode of constant does not agree with mode of Factor.
c. Invalid characters appear in constant (e.g., non-digits if the constant is numeric).
d. Sign is omitted from binary or packed constant.

SRT – C:?NO HEADER [RMS-Status-Code]

a. First record of specification file is not an H specification.

SRT – C:?INCORRECT SEQUENCE [RMS-Status-Code]

a. Numeric record sequence is lower than sequence previously encountered.
b. No valid data specification appears when keys are to be stripped from output.
c. Record specification after "include-all" ("include-all" should be last).
d. Key specifications appear after data specifications.

SRT – C:?NO ALTSEQ [RMS-Status-Code]

a. Specification for alternate collation entered in Header column 26 but no ALTSEQ Specifications follow.

SRT – C:?TOO MANY SPECIFICATIONS [RMS-Status-Code]

a. Number of specifications for a particular type of record have overflowed the buffer space.

NOTE

SORT reserves a buffer space for storage of a table based on the input specifications and used to control the processing of each record type. This space should be ample for all situations to make the error unlikely other than in very exceptional circumstances.

APPENDIX C
TRAX I/O ERROR CODES

The following I/O error codes are return to TRAX Tasks:

Mnem.	Dec.	Octal	
.BAD	-1	377	Bad parameters
.IFC	-2	376	Invalid function code
.DNR	-3	375	Device not ready
.VER	-4	374	Parity error on device
.ONP	-5	373	Hardware option not present
.SPC	-6	372	Illegal user buffer
.DNA	-7	371	Device not attached
.DAA	-8	370	Device already attached
.DUN	-9	367	Device not attachable
.EOF	-10	366	End-Of-file detected
.EOV	-11	365	End-of volume detected
.WLK	-12	364	Write attempted to looked unit
.DAO	-13	363	Data overrun
.SRE	-14	362	Send/receive failure
.ABO	-15	361	Request terminated
.PRI	-16	360	Privilege violation
.RSU	-17	357	Shareable resource in use
.OVR	-18	356	Illegal overlay request
.BYT	-19	355	Odd byte count (or virtual address)
.BLK	-20	354	Logical block number too large
.MOD	-21	353	Invalid UDC module number
.CON	-22	352	UDC connect error
.NOD	-23	351	System dynamic memory exhausted
.DFU	-24	350	Device full
.IFU	-25	347	Index file full
.NSF	-26	346	No such file
.LCK	-27	345	Locked from read/write access
.HFU	-28	344	File header full
.WAC	-29	343	Accessed for write
.CKS	-30	342	File header checksum failure
.WAT	-31	341	Attribute control list format error
.RER	-32	340	File processor device read error
.WER	-33	337	File processor device write error
.ALN	-34	336	File already accessed on LUN

.SNC	-35	335	File ID, file number check
.SQC	-36	334	File ID, sequence number check
.NLN	-37	333	No file accessed on LUN
.CLO	-38	332	File was not properly closed
.NBF	-39	331	No buffer space available for file
.RBG	-40	330	Illegal record size
.NBK	-41	327	File exceeds space allocated, no blocks
.ILL	-42	326	Illegal operation on file descriptor block
.BTP	-43	335	Bad record type
.RAC	-44	324	Illegal record access bits set
.RAT	-45	323	Illegal record attributes bits set
.RCN	-46	322	Illegal record number-too large
	-47		(not used)
.2DV	-48	320	Rename-2 different devices
.FEX	-49	317	Rename - a new file name already in use
.BDR	-50	316	Bad directory file
.RNM	-51	315	Cannot rename old file system
.BDI	-52	314	Bad directory syntax
.FOP	-53	313	File already open
.BDV	-55	311	Bad device name
.BBE	-56	310	Bad block on device
.DUP	-57	307	Enter-duplicate entry in directory
.STK	-58	306	Not enough stack space (FCS or FCP)
.FHE	-59	305	Fatal hardware error on device
.NFI	-60	304	File ID was not specified
.ISQ	-61	303	Illegal sequential operation
.EOT	-62	302	End-of tape detected
.BVR	-63	301	Bad version number
.BHD	-64	300	Bad file header
.OFL	-65	277	Device offline
.BCC	-66	276	Block check or CRC error
	-67		(not used)
.NNN	-68	274	No such node
.NFW	-69	273	Path lost to partner
.BLB	-70	272	Bad logical buffer
.TMM	-71	271	Too many outstanding messages
.NDR	-72	270	No dynamic space available
.CNR	-73	267	Connection rejected
.TMO	-74	266	Time out on request
.EXP	-75	265	File expiration date not reached
.BTF	-76	264	Bad tape format
.NNC	-77	263	Not ANSI "D" format byte count
.NNL	-78	262	Not a network LUN
.NLK	-79	261	Task not linked to specified ICS/ ICR interrupts

.NST	-80	260	Specified task not installed
.FLN	-81	257	Device offline when offline request was issued
.IES	-82	256	Invalid escape sequence
.PES	-83	255	Partial escape sequence
.ALC	-84	254	Allocation failure
.ULK	-85	253	Unlock error

APPENDIX D
RMS COMPLETION STATUS CODES

This appendix describes completion status codes that can be returned by RMS-11 to your program.

All RMS-11 file and record operations return a completion status code into the status field (STS) of the control block (i.e., FAB or RAB) associated with the operation. A symbolic name is defined for each such code. The symbolic names for successful completion status codes take the following form:

SU$xxx

where

xxx is a mnemonic value describing the successful operation.

Symbolic names for error completion status codes take the form:

ER$xxx

where

xxx is a mnemonic value representing the reason the operation failed.

For certain error conditions, RMS-11 uses the status value (STV) field to communicate additional information to your program. The tables in this appendix list all instances in which a particular symbolic value in the STS field indicates the presence of further information in the STV field. A limited number of severe error conditions cause RMS-11 to invoke a fatal error crash routine. Section D.1 of this appendix describes these conditions and the crash routine itself.

The sections that follow describe, respectively, successful completion status codes, error completion status codes, and the RMS-11 fatal error crash routine.

D.1 SUCCESSFUL COMPLETION STATUS CODES

Table D-1 describes successful completion status codes returned by RMS-11 routines.

Table D-1 Successful Completion Status Codes

Symbolic Name	Decimal Value	Description.
SU$SUC	1	Operation successful.
SU$DUP	2	A record written into an indexed file as a result of a $PUT or $UPDATE operation contains at least one key value that was already present in another record.
SU$IDX	3	During a $PUT or $UPDATE operation on an indexed file, the record was successfully written. The record can be subsequently retrieved but RMS-11 was not able to optimize the structure of the index at the time the record was inserted. Several indirections will occur, therefore, on retrieval. In some instances, RMS-11 may also return an error code (e.g., ER$RLK) in the STV field of the control block.
SU$RRV	4	During a $PUT or $UPDATE operation on an indexed file, the record was successfully written. However, RMS-11 was unable to update one or more Record Retrieval Vectors (RRVs) and the records associated with the RRVs cannot be retrieved using alternate indexes or RFA addressing mode.

D.2 ERROR COMPLETION STATUS CODES

Table D-2 describes error completion status codes returned by RMS-11 routines.

Table D-2 Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$ABO	177760	-16	ER$STK or ER$MAP	Operation aborted: out of stack save area or in core data structures corrupted.
ER$ACC	177740	-32	Kernel Error code	Kernel file system could not access the file.
ER$ACT	177720	-48		File activity precludes action (e.g., attempting to close a file with outstanding asynchronous record operation).

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$AID	177700	-64	XAB address	Bad area identification number (AID) field in allocation XAB (i.e., out of sequence).
ER$ALN	177660	-80	XAB address	Illegal value in alignment boundary type (ALN) field allocation XAB.
ER$ALQ	177640	-96	(XAB address)	Value in allocation quantity (ALQ) field in FAB (or allocation XAB) exceeds maximum or, during an explicit $EXTEND operation, equals zero.
ER$ANI	177620	-112		Records in a file on ANSI labeled magnetic tape are variable length but not in ANSI D format.
ER$AOP	177600	-128	XAB address	Illegal value in allocation options (AOP) field in allocation XAB.
ER$AST	177560	-144		Invalid operation at AST level: attempting to issue a synchronous operation from an asynchronous record operation completion routine.
ER$ATR	177540	-160	Kernel Error code	Read error on file header attributes.
ER$ATW	177520	-176	Kernel Error code	Write error on file header attributes.
ER$BKS	177500	-192		Bucket size (BKS) field in FAB contains value exceeding maximum.
ER$BKZ	177460	-208	XAB address	Bucket size (BKZ) field in allocation XAB contains value exceeding maximum.
ER$BLN	177440	-224		Block length (BLN) field in a FAB or RAB is incorrect.

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$BOF	177430	-232		Beginning of file detected on $SPACE operation to magnetic tape file.
ER$BPA	177420	-240		Private buffer pool address not a double word boundary.
ER$BPS	177400	-256		Private buffer pool size not a multiple of 4.
ER$BUG	177360	-272		Internal error detected in RMS-11 (refer to Section D.4 of this Appendix); no recovery possible; contact a Software Specialist.
ER$CCR	177340	-288		Can't connect RAB (i.e., only one record access stream permitted for sequential files).
ER$CHG	177320	-304		$UPDATE attempting to change a key field that does not have the change attribute.
ER$CHK	177300	-320		Index file bucket check-byte mismatch. The bucket has been corrupted. No recovery possible for the bucket.
ER$COD	177240	-352	XAB address	Invalid COD field in XAB or XAB type is illegal for the organization or operation.
ER$CRE	177220	-368	Kernel Error code	Kernel file system could not create file.
ER$CUR	177200	-384		No current record: operation not immediately preceded by a successful $GET or $FIND.
ER$DAC	177160	-400	Kernel Error code	Kernel file system deaccess error during $CLOSE

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$DAN	177140	-416	XAB address	Invalid area number in DAN field of key definition XAB.
ER$DEL	177120	-432		Record accessed by RFA access mode has been deleted.
ER$DEV	177100	-448		1. Syntax error in device name. 2. No such device. 3. Inappropriate device for operation (e.g., attempting to create an indexed file on magnetic tape).
ER$DIR	177060	-464		Syntax error in directory name.
ER$DME	177040	-480		Dynamic memory exhausted: insufficient space in central space pool or private buffer pool.
ER$DNF	177020	-496		Directory not found.
ER$DNR	177000	-512		Device not ready.
ER$DPE	176770	-520	Kernel Error code	Device positioning error.
ER$DUP	176740	-544		Duplicate key detected, duplicates allowed attribute not set for one or more key fields.
ER$ENT	176720	-560	Kernel Error code	Kernel file system enter function failed.
ER$ENV	176700	-576		Environment error: operation or file organization not specified in ORG$ macro.
ER$EOF	176660	-592		End of file.
ER$ESS	176640	-608		Expanded string area in NAM block too short.
ER$EXP	176630	-616		File expiration date not reached.

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$EXT	176620	-624	Kernel Error code	File extend failure.
ER$FAB	176600	-640		Not a valid FAB: BID field does not contain FB$BID. Refer to Section A.3 of this Appendix.
ER$FAC	176560	-656		1. Record operation attempted was not declared in FAC field of FAB at open time. 2. Invalid contents in FAC field. 3. FB$PUT not present in FAC for $CREATE operation.
ER$FEX	176540	-672		File already exists (attempted $CREATE operation).
ER$FID	177530	-680		Invalid file id.
ER$FLG	176520	-688	XAB address	Invalid combination of values in FLG field of key definition XAB (e.g., no duplicates and keys can change).
ER$FLK	176500	-704		File locked by another user - - you cannot access the file because your sharing specification cannot be met.
ER$FND	176460	-720	Kernel Error code	Kernel file system Find function failed.
ER$FNF	176440	-736		File not found.
ER$FNM	176420	-752		Syntax error in file name.
ER$FOP	176400	-768		Invalid file options.
ER$FUL	176360	-784		Device full: can't create or extend file.
ER$IAN	176340	-800	XAB address	Invalid area number in IAN field of key definition XAB.

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$IDX	176320	-816		Index not initialized (this code can only occur in the STV field when STS contains ER$RNF).
ER$IFI	176300	-832		Invalid IFI field in FAB.
ER$IMX	176260	-848	XAB address	Maximum number (254) of key definition or allocation XABs exceeded or multiple summary, protection, or date XABs present during operation.
ER$INI	176240	-864		$INIT or $INITIF macro call never issued.
ER$IOP	176220	-880		Illegal operation; examples include: 1. Attempting a $TRUNCATE operation to a non-sequential file. 2. Attempting an $ERASE or $EXTEND operation to a magnetic tape file. 3. Issuing a block mode operation (e.g., $READ or $WRITE) to a stream not connected for block operations. 4. Issuing a record operation (e.g., $GET, $PUT) to a stream connected for block mode operations.
ER$IRC	176200	-896		Illegal record encountered in sequential file: invalid count field.
ER$ISI	176160	-912		Invalid internal stream identifier (ISI) field in RAB (field may have been altered by user) or $CONNECT never issued for stream.
ER$KBF	176140	-928		Key buffer address (KBF) field equals 0.

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$KEY	176120	-944		Record identifier (i.e., the 4-byte location addressed by KBF) for random operation to relative file is 0 or negative.
ER$KRF	176100	-960		Invalid key of reference (KRF) in RAB: 1) As input to random $GET or $FIND operation, or 2) As input to $CONNECT or $REWIND (in this case, ER$KRF is returned for the first record operation following the $CONNECT or $REWIND.
ER$KSZ	176060	-976		Key size equals zero or too large (indexed file) or not equal to 4 (relative file).
ER$LAN	176040	-992	XAB address	Invalid area number in LAN field of key definition XAB.
ER$LBL	176020	-1008		Magnetic tape is not ANSI labeled.
ER$LBY	176000	-1024		Logical channel busy.
ER$LCH	175760	-1040		Invalid value in logical channel number (LCH) field of FAB.
ER$LEX	175750	-1048	XAB address	Attempt to extend an area containing an UNUSED extent.
ER$LOC	175740	-1056	XAB address	Invalid value in LOC field of allocation XAB.
ER$MAP	175720	-1072		In core data structures (e.g., I/O buffers) corrupted. This code can only occur in the STV field when STS contains ER$ABO. Refer also to Section D.4 of this Appendix.
ER$MKD	175700	-1088	Kernel Error code	Kernel file system could not mark file for deletion.

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$MRN	175660	-1104		1. Maximum record number field contains a negative value during $CREATE of relative file. 2. Record identifier (pointed to by KBF) for random operation to relative file exceeds maximum record number specified when file created.
ER$MRS	175640	-1120		Maximum record size (MRS) field contains 0 during $CREATE operation and: 1. Record Format is fixed, or 2. File organization is relative.
ER$NAM	175620	-1136		Odd address in Name Block address (NAM) field in FAB on $OPEN, $CREATE, or $ERASE.
ER$NEF	175600	-1152		Not at end-of-file: attempting a $PUT operation to a sequential file when stream is not positioned to EOF.
ER$NID	175560	-1168		Can't allocate internal index descriptor: insufficient room in space pool while attempting to open an indexed file.
ER$NPK	175540	-1184		No primary key definition XAB present during $CREATE of indexed file.
ER$ORD	175500	-1216	XAB address	XABs in chain not in correct order: 1. Allocation or key definition XABs not in ascending (or densely ascending) order. 2. XAB of another type intervenes in key definition or allocation XAB sub-chain.

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$ORG	175460	-1232		Invalid value in file organization (ORG) field of FAB.
ER$PLG	175440	-1248		Error in file's prologue: file is corrupted and must be reconstructed.
ER$POS	175420	-1264	XAB address	Key position (POS) field in key definition XAB contains a value exceeding maximum record size.
ER$PRM	175400	-1280	XAB address	File header contains bad date and time information (retrieved by RMS-11 because a date and time XAB is present during a $OPEN or $DISPLAY operation); file may be corrupted.
ER$PRV	175360	-1296		Privilege violation: access to the file denied by the operating system.
ER$RAB	175340	-1312		Not a valid RAB: BID field does not contain RB$BID. Refer to Section D.4 of this Appendix.
ER$RAC	175320	-1328		1. Illegal values in record access mode (RAC) field of RAB. 2. Illogical value in RAC field (e.g., RB$KEY with a sequential file).
ER$RAT	175300	-1344		1. Illegal values in record attributes (RAT) field of FAB during $CREATE. 2. Illogical combination of attributes (e.g., FB$CR and FB$FTN) in RAC field during $CREATE.

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$RBF	175260	-1360		Record Address (RBF) field in RAB contains an odd address (block mode access only).
ER$RER	175240	-1376	Kernel Error code	File read error.
ER$REX	175220	-1392		Record already exists: during a $PUT operation in random mode to a relative file, an existing record found in the target record position.
ER$RFA	175200	-1408		Invalid RFA in RFA field of RAB during RFA access.
ER$RFM	175160	-1424		1. Invalid record format in RFM field of FAB during $CREATE. 2. Specified record format is illegal for file organization.
ER$RLK	175140	-1440		Target bucket locked by another task or another stream in the same program.
ER$RMV	175120	-1456	Kernel Error code	Kernel file system Remove function failed.
ER$RNF	175100	-1472	(ER$IDX)	Record identified by KBF/KSZ fields of RAB for random $GET or $FIND operation does not exist in relative or indexed file (for indexed files only, STV may contain ER$IDX). Record may never have been written or may have been deleted.
ER$RNL	175060	-1488		$FREE operation issued but no bucket was locked by stream.
ER$ROP	175040	-1504		Record options (ROP) field contains illegal values or illogical combination of values.

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$RPL	175020	-1520	Kernel Error code	Error while reading prologue.
ER$RRV	175000	-1536		Invalid RRV record encountered in indexed file; file may be corrupted.
ER$RSA	174760	-1552		Record stream active, i.e., in asynchronous environment, attempting to issue a record operation to a stream that has a request outstanding.
ER$RSZ	174740	-1568		Record size specified in RSZ of RAB during $PUT or $UPDATE is invalid: 1. RSZ equals zero. 2. RSZ exceeds maximum record size (MRS) specified when file created. 3. RSZ not equal to size of Current Record for $UPDATE operation to a sequential file on disk. 4. RSZ does not equal MRS (for fixed format records).
ER$RTB	174720	-1584	Actual record size	Record too big for user's buffer: RMS-11 could not move entire record retrieved by $GET operation to user work area (UBF/USZ). Note that this error does not destroy the current context of the stream. Rather, the stream's context is updated as if the operation had been completely successful.
ER$SEQ	174700	-1600		During $PUT operation, key of record to be written is not equal to or greater than key of previous record (and RAC field contains RB$SEQ).

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$SHR	174660	-1616		Illogical value in SHR field of FAB (e.g., FB$WRI specified for sequential file).
ER$SIZ	174640	-1632	XAB address	Invalid SIZ field in key definition XAB during $CREATE (e.g., specified size exceeds maximum record size).
ER$STK	174620	-1648		During asynchronous record operation, RMS-11 has found that the stack is too big to be saved (this code can only occur in the STV field when STS contains ER$ABO).
ER$SYS	174600	-1664	Directive or QIO status code	System directive error.
ER$TRE	174560	-1680		Index tree error: indexed file is corrupted.
ER$TYP	174540	-1696		Syntax error in file type (e.g., more than 3 characters specified).
ER$UBF	174520	-1712		Invalid address in UBF field of RAB: 1. UBF contains 0, or 2. UBF not word aligned (for block mode access only).
ER$USZ	174500	-1728		Invalid USZ field in RAB (i.e., USZ contains 0).
ER$VER	174460	-1744		Syntax error in file version number.
ER$VOL	174440	-1760	XAB address	Invalid VOL field in allocation XAB (i.e., VOL does not contain 0).
ER$WER	174420	-1776	Kernel Error code	File write error.
ER$WLK	174410	-1784		Device is write locked.

Table D-2 (Cont.) Error Completion Status Codes

Symbolic Value	Octal Value	Decimal Value	STV	Description
ER$WPL	174400	-1792	Kernel Error code	Error while writing prologue.
ER$XAB	174360	-1808	(XAB address)	XAB field in FAB (or NXT field in XAB) contains an odd address.

D.3 FATAL ERROR CRASH ROUTINE

RMS-11 issues a BPT instruction whenever it encounters inconsistent internal FAB or RAB). This action is taken only when RMS-11 cannot continue processing, since to do so might cause damage to user files or the user's task image. As an example, when the problem is caused by an invalid FAB or RAB, RMS-11 cannot return an error status code in STS since it has recognizable user control block to work with.

The BPT instruction generated as a result of fatal errors is in the R0RMSA module of RMS-11. The following is the state of the general registers at the time this instruction is issued:

R0 = RMS-11 error code
R1 = Entry SP value
R2 = Entry return PC
R3 = Address of system impure area

General registers R1 and R2 are always valid if the crash routine is invoked by a fatal user call error. When the crash routine in invoked by inconsistent internal conditions, the contents of general registers R1 is R2 may be meaningless if RMS-11 was executing an asynchronous RAB operation.

The following subsections summarize, respectively, the fatal user call errors and the RMS-11 inconsistent internal conditions that can cause invocation of the fatal error crash routine.

D.4 FATAL USER CALL ERRORS

When the fatal error crash routine is invoked because of a user call error, general register R0 contains one of the following error codes:

- ER$FAB
- ER$RAB

These error codes indicate that the user called RMS-11 using a control block that was not a valid FAB (for file operations such as $OPEN, $CREATE, etc) or RAB (for record operations such as $CONNECT, $GET, $PUT, etc.). This condition can occur for any one of the following reasons:

1. ;The address of the FAB or RAB is 0.
2. The address of the FAB or RAB is odd.
3. ;The control block's BID field does not contain the proper block identifier code (i.e., FB$BID for FABs and RB$BID for RABs).

D.5 RMS-11 INCONSISTENT INTERNAL CONDITIONS ERRORS

When the crash routine is invoked because of RMS-11 inconsistent internal conditions, general register R0 contains one of the following error codes:

- ER$BUG
- ER$MAP

These error codes indicate internal problems with RMS-11 and are considered fatal. They can be caused by improper coding by the user (e.g., destroying some internal RMS-11 data base), but are also used to detect RMS-11 bugs. When one of the above error codes is encountered, the user should provide, if possible, the following information to DEC with an SPR:

1. The contents of the general registers.
2. The first ten words, at a mininum, or all words upon the system stack.
3. The operation the program was performing (e.g., $OPEN, $GET, $PUT).
4. The organization of the file being processed.
5. A load map of the task.
6. If running on TRAX, a post-mortem dump.

INDEX

TRAX Support
Environment
User's Guide
AA-D331A-TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
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

Software Documentation
146 Main Street ML 5-5/E39
Maynard, Massachusetts 01754