August 1978

The manual describes the VAX/VMS command language, DCL. It provides detailed reference information and examples of all non-privileged commands available to general users.

VAX/VMS
Command Language
User's Guide

Order No. AA-D023A-TE

SUPERSESSION/UPDATE INFORMATION:	This is a new document for this release.
OPERATING SYSTEM AND VERSION:	VAX/VMS V01
SOFTWARE VERSION:	VAX/VMS V01

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
VAX	VMS	SBI
DECnet	IAS	

CONTENTS

FIGURES

PART I

PART II

TABLES

PART I

PART II

PREFACE

0.1 Manual Objectives

This manual describes the VAX/VMS command language, DCL (DIGITAL Command Language), and provides usage and reference information on the command language.

0.2 Intended Audience

This manual is intended for all users of the VAX/VMS operating system, including applications programmers, system programmers, operators, and managers.

The VAX/VMS Primer is a tutorial manual that introduces the VAX/VMS operating system and the use of the command language. General users who are not familiar with interactive computer systems should read the Primer before using this command language user's guide.

The VAX/VMS Summary Description introduces operating system concepts of general interest. It is recommended that system programmers and managers be familiar with the material in the summary description before using this command language user's guide.

0.3 Structure of this Document

The manual has two parts:

Part I. Using the Command Language

This part is tutorial; it provides an overview of the command language and operating system concepts. Part I contains six chapters:

- Chapter 1, "Overview" describes how to access the system and enter commands. At the end of Chapter 1, on colored pages, are tables that summarize the DCL commands in functional categories.
- Chapter 2, "File Specifications and Logical Names" explains the device and file naming conventions of the operating system and describes how to assign and use logical names to refer to devices and files in commands and programs.
- Chapter 3, "Disk and Tape Volumes" discusses concepts related to accessing files on disk and tape volumes, and provides examples of initializing and using disks and tapes.

- Chapter 4, "Programming with VAX/VMS" gives an overview of the program development tools provided by DCL commands and describes the environment in which programs execute.

- Chapter 5, "Command Procedures and Batch Jobs" explains how to create command procedures to execute sequences of commands, how to submit command procedures for execution as batch jobs, and how to use some of the language features available in DCL to construct complex procedures.

- Chapter 6, "Grammar Rules" describes the syntax of the command language and defines the rules for entering commands, command parameters and qualifiers, numeric values and expressions, and character data.

Part II. Command Descriptions

This part contains detailed descriptions of each command. The commands are listed in alphabetical order, with the command name appearing at the top of the first and every page of the individual command description.

0.4 Associated Documents

See the VAX-11 Information Directory for a description of related documents and to obtain the order numbers of manuals referred to in this book.

0.5 Conventions used in this Document

(RET)	A symbol with a one- to three-character abbreviation indicates that you press a key on the terminal, for example, (RET) or (ESC) .
CTRL/x	The phrase CTRL/x indicates that you must press the key labeled CTRL while you simultaneously press another key, for example CTRL/C, CTRL/Y, CTRL/O. In examples, this control key sequence is shown as ^x, for example ^C, ^Y, ^O, because that is how the system echoes control key sequences.
$ SHOW TIME 05-jun-1978 11:55:22	Command examples show all output lines or prompting characters that the system prints or displays in black letters. All user-entered commands are shown in red letters.
$ FORTRAN MYFILE $ LINK MYFILE $ RUN MYFILE	Examples showing the contents of files are enclosed in boxes.
$ TYPE MYFILE.DAT . .	Vertical series of periods, or ellipses mean that not all of the data that the system would display in response to the particular command is shown; or, that not all the data a user would enter is shown.

file-spec,...	Horizontal ellipses indicate that additional parameters, values, or information can be entered.
[logical-name]	Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

PART I

USING
THE COMMAND LANGUAGE

CHAPTER 1

OVERVIEW

The DCL command language provides VAX/VMS users with an extensive set of commands for:

- Interactive program development
- Device and data file manipulation
- Interactive and batch program execution and control

The following sections show how to access the system and describe how the system processes commands interactively. These sections are followed by tables, on colored pages, that describe all the commands available to general users. The commands are listed in functional categories to provide a quick overview of VAX/VMS capabilities.

1.1 ACCESSING THE SYSTEM

When a terminal is physically connected to the system, you can get its attention and signal that you want to begin a terminal session by pressing the RETURN or CTRL/Y key. The system responds by prompting you for your user name. After you enter your user name, the system prompts you to enter your password. For example:

```
(RET)
Username: PATTI (RET)
Password:
```

When you enter the password, the system does not echo it, that is, the system accepts the input line, but does not display it on the terminal.

During this sequence, known as login, the system validates that you are authorized to use the system. When the login is complete, the system may display messages telling you about the system. Finally, the system indicates that it is ready to accept commands by displaying the prompting character, a dollar sign ($), as shown below:

```
      WELCOME TO VAX/VMS RELEASE 1
$
```

1.2 YOUR COMMAND ENVIRONMENT

When you have logged into the system, and the system is ready to accept commands, the system defines the environment within which it responds to your requests. This environment is called a process.

Associated with your process are various default characteristics, among which are the following:

- An account number, which your installation uses to keep track of the computer resources used.
- A user identification code (UIC), which provides you with a group number and a member number within the group. Other users can have the same group number. Within groups, users are allowed to share files or system resources more freely.
- A default disk device and directory name, which the system uses to locate and catalog files that you create or use.
- Default devices for the input, output, and error streams of the process, from which the system reads command and program input and to which the system writes messages.
- A set of privileges and resource quotas that define what system resources or system functions you, or programs you execute, will be allowed to use. Tables 1-7 and 1-8 at the end of this chapter summarize the specific privileges and quotas the system defines.
- A command interpreter -- the operating system component responsible for reading and translating your typed-in requests. This manual describes only the DCL command interpreter.

The system obtains the characteristics unique to your process from the user authorization file. The user authorization file is a list of all users who can access the system; the system manager or operator maintains this file.

1.3 ENTERING COMMANDS

Commands consist of English-language names (generally verbs) that describe what you want the system to do. Commands can optionally contain qualifiers and parameters. Command qualifiers modify a command: they provide the system with additional information on how to execute the command. Command parameters describe the object of the command: in some cases, a parameter is a keyword; in other cases, it is a file or a device to manipulate or a program to execute.

The following example shows a PRINT command, as it would appear on a terminal:

```
$ PRINT/COPIES=2 MYFILE.DAT (RET)          ! Print the file
  Job 210 entered on queue LPB0:
$
```

Notes:

$	The system prompt for command input; a dollar sign means that the command interpreter is ready for you to type a command.
PRINT	The command name, requesting the system to queue a file for printing on the system printer.

/COPIES=2
: A qualifier to the PRINT command, requesting that two copies of the specified file be printed. Qualifiers are always preceded with slash characters (/), and, if they require a value (as in this example), the qualifier name is separated from the value with either an equals sign (=) or a colon (:).

MYFILE.DAT
: The command parameter, naming the file to be printed. In this command, as for many DCL commands, the parameter is a file specification. At least one blank space must always immediately precede a parameter.

RET
: The carriage return. Pressing this key after entering a full command line terminates the command input; the system begins processing the command. Note that examples in this manual do not explicitly show the RET following commands; you can assume that all lines shown in examples must be terminated with RET unless stated otherwise.

Job 210 entered on queue LPB0:
: A message from the PRINT command, indicating that the command completed successfully; the command interpreter gave the print job an identification number and queued it to the system printer named LPB0.

$
: The next $ prompt, indicating that the PRINT command has completed successfully; that is, two copies of the file MYFILE.DAT were queued for printing and the system is ready to accept another command.

! Print the file
: A comment. The command interpreter ignores any data beginning with an exclamation point.

Chapter 6, "Grammar Rules" contains complete details on the syntax rules for entering commands, parameters, and qualifiers, including:

- Continuing commands on more than one line
- Shortening command and other keyword names to four or fewer characters
- Specifying values for qualifiers

1.4 COMMAND PROMPTING

When you enter a command at the terminal, you need not enter the entire command all at once. If you enter a command that requires parameters and do not specify any parameters, the command interpreter prompts you for all remaining parameters, including optional parameters. For example:

```
$ PRINT/COPIES=2
$_File:      MYFILE.DAT
  Job 211 entered on queue LPA1
```

In this example, no parameter is entered, so the system prompts for a file specification. A line beginning with $_ indicates that the system is in prompt mode.

When you are prompted for an optional parameter, you can press (RET) to complete the command sequence without specifying the parameter, as the following example illustrates:

```
$ ALLOCATE
$_Device:    DM:
$_Log_Name: (RET)
  _DMB1: ALLOCATED
```

In the above example, the ALLOCATE command is entered with no parameters; the command interpreter begins prompting for all parameters. The second prompt, $_Log_Name:, is for an optional logical name parameter. A null entry, signaled by <RET>, terminates the command entry.

1.5 SYSTEM MESSAGES

When you enter a command incorrectly, the command interpreter issues a descriptive error message telling you what was wrong. For example, if you specify more than one parameter for a command that accepts a single parameter, you receive the message:

```
%DCL-W-MAXPARM, maximum parameter count exceeded
```

You must then retype the command.

Other error messages may occur during execution of a command. These messages can indicate such errors as a nonexistent file or a conflict in qualifiers.

Not all messages from the system indicate errors; some messages are informative, or merely warn you of a particular condition.

Messages have the general format:

```
%XXX-L-CODE, text
```

where XXX is a mnemonic for the operating system facility or the program issuing the message, L is a severity level indicator (S for success, I for informational, W for warning, E for error, and F for fatal), and CODE is a shorthand code for the message text.

Because the messages are descriptive, you can usually tell what you need to do when you issue the command again.

The VAX/VMS System Messages and Recovery Procedures Manual lists the system messages and describes what you can do to correct an error.

1.6 THE HELP COMMAND

When you are using the system, you may not always have this user's guide available at your terminal when you need a summary of the format of a particular command or a list of its valid qualifiers. The HELP command provides you with this information. For example, if you type the command:

```
$ HELP PRINT
```

The system responds by displaying an abstract of the PRINT command and keywords you can enter as parameters to the HELP command to obtain more information about PRINT. If you enter the following:

```
$ HELP PRINT QUALIFIERS
```

The HELP command displays a description of each PRINT command qualifier.

1.7 EXTENDING THE COMMAND LANGUAGE

Beyond the normal syntax and acceptable command names and parameters, VAX/VMS lets you define synonyms for command names, and "create" commands by writing command procedures.

1.7.1 Synonyms for DCL Commands

You can create synonyms by using symbolic names. You define symbolic names with assignment statements that equate a symbol name to a numeric value or a character string. For example, you could define a symbol name to be equated to a real DCL command name as follows:

```
$ LIST := DIRECTORY
```

For the remainder of the terminal session, when you enter the symbol LIST as the first word on a command line, the command interpreter substitutes the symbol with the string DIRECTORY and executes the DIRECTORY command.

A symbol can also contain a portion of a command, for example:

```
$ PDEL := DELETE SYS$PRINT/ENTRY=
```

This assignment statement equates the symbol name PDEL with the command necessary to delete an entry from the printer queue, except that the required value for the /ENTRY qualifier is omitted. When you use this symbol as a command synonym, you must also type the job identification number assigned to the entry, as shown below:

```
$ PDEL 210
```

When the command interpreter processes this line, it substitutes the symbol PDEL with its current value and then executes the command:

```
DELETE SYS$PRINT/ENTRY=210
```

For additional information on defining symbols for DCL command synonyms, see the description of the = (Assignment Statement) command in Part 2.

1.7.2 Command Procedures

A command procedure is a file that contains a sequence of DCL commands, some of which can conditionally control the execution of the procedure. By placing sets of frequently used commands and/or qualifiers in command procedures, you can construct command language "programs" from DCL commands. After you create a command procedure,

you can execute all the commands in it with a single command. For example, suppose a procedure named TESTALL.COM contains the command lines:

```
$ RUN WEEKCALC
$ RUN TIMECARDS
$ PRINT WEEKCALC.OUT,TIMECARDS.OUT
```

You can execute the three commands in this file by entering:

```
$ @TESTALL
```

You can also use the SUBMIT command to request the system to process a command procedure as a batch job, so that your terminal remains free for interactive work.

Chapter 5, "Command Procedures and Batch Jobs" provides details on using the command interpreter's symbolic capabilities and on developing and using command procedures.

1.8 THE FILE SYSTEM

Using DCL commands, you can create, access, and update data files and programs. The VAX-11 Record Management Services (RMS) provide the access and control capabilities that are called by the DCL commands you request.

You can also define and access files from within your programs by using RMS or by using the input/output services of the VAX/VMS operating system directly.

Chapter 2, "File Specifications and Logical Names" describes how to identify and refer to files. It also describes the directory structure of the files on disk volumes and explains how to create a hierachy of directories to catalog and maintain files.

Table 1-3, at the end of this Chapter, lists the DCL commands that provide file manipulation functions. Each of these commands is described in detail in Part II of this manual. For a list of additional manuals that contain information on file services and utilities, see the VAX-11 Information Directory.

1.9 TERMINAL CHARACTERISTICS

Your terminal keyboard is the means by which you communicate with the command interpreter or enter data solicited by a command or program. Although a terminal keyboard is similar to that of a typewriter, there are several differences between the way a typewriter and your terminal accept and display data that you type. These differences include:

- The interpretation of uppercase and lowercase alphabetic characters
- The type-ahead buffer
- Special function keys

When you type input lines to the command interpreter from a terminal that displays lowercase letters, the command interpreter translates all lowercase characters to uppercase before it executes a command, unless you enter character strings enclosed in quotation marks.

After you enter a command, and while the command interpreter is responding to it or executing it, your keyboard is not locked; that is, you can continue typing. The system saves what you type during this time in a special buffer called the type-ahead buffer. It does not, however, echo what you type. When the command currently being executed completes, the command interpreter displays what you have typed.

If you typed a full command line, including a <RET>, the command interpreter executes the command after displaying it. Otherwise, it displays what you have typed and waits for you to enter the rest of the line. If you have entered more than one complete line, it displays each line just before executing it.

Some of the special terminal function keys provide line-editing functions, so you can correct or cancel what you type before passing it to the command interpreter. Other keys can interrupt system processing. Some of the line-editing keys are described in the next section. Table 1-1 (at the end of Section 1.9.2) lists all of the terminal function keys and describes their use.

1.9.1 Special Terminal Function Keys

As you type commands or program data at the terminal, you can take advantage of several keys that let you make changes to lines as you type them. Thus, if you make a mistake, you can correct it before pressing the return key.

1.9.1.1 **Deleting Characters** - The (DEL) (DELETE) key backspaces over the most recently entered character and deletes it. For example:

```
$ PRO (DEL) INT
```

After incorrectly typing the letter "O", you can delete it by pressing (DEL) and then continue your input by typing "INT". On a hardcopy terminal, the letters deleted are displayed between backslash characters so you can see what is being deleted. For example:

```
$ PRO\O\INT
```

On a display terminal, pressing (DEL) actually erases the character from the screen and moves the cursor backwards. Note that the key that performs the delete function is marked RUBOUT on some terminals.

1.9.1.2 **Deleting Lines** - To cancel an entire line, you can use the CTRL/U key sequence. To use the CTRL key, you must press it at the same time you press the other letter (in this case, U). For example, if you make many mistakes on a particular line and want to cancel the line and reenter it, use CTRL/U as shown below:

```
$ PRNT/CIPY=2^U
$ PRINT/COPIES=2 MYFILE.DAT
```

When you cancel a line with CTRL/U, the system ignores the line and prompts you for the next line.

1.9.1.3 **Canceling Commands** - If you are entering commands in response to prompts and want to discard the entire command you have entered so far (not just the most recently entered line), use CTRL/C or CTRL/Y, as shown below:

```
$ PRINT/COPIES=3
$_File:     MYFILE.DAT ^C
```

In this example, the PRINT command was entered correctly, and the system prompted for the name of a file to print. However, while entering the file name, the user decided not to enter the command at all, and used CTRL/C to discard the entire command.

1.9.2 Setting Terminal Characteristics

There are many types of terminal, and each has its own operating characteristics. You can change some of these characteristics, based on your requirements, with the SET TERMINAL command. To determine the current characteristics of your terminal, issue the SHOW TERMINAL command, as shown below:

```
$ SHOW TERMINAL
```

Each of the parameters displayed can be changed with corresponding parameters for the SET TERMINAL command. For example:

```
$ SET TERMINAL/LOWERCASE
```

After you issue this command, all lowercase alphabetic characters are displayed in lowercase (if your terminal is capable of displaying lowercase letters).

Table 1-1
Terminal Function Keys

Key	Function
RETURN	Carriage return; transmits the current line to the system for processing. (On some terminals, the RETURN key is labeled CR.) Before a terminal session, initiates login sequence.
Control keys	Define functions to be performed when the CTRL key and another key are pressed simultaneously. All CTRL/x key sequences are echoed on the terminal as ^x.

(continued on next page)

Table 1-1 (Cont.)
Terminal Function Keys

Key	Function
CTRL/C [1]	During command entry, cancels command processing.
	Before a terminal session, initiates login sequence.
CTRL/I	Duplicates the function of the TAB key.
CTRL/K	Advances the current line to the next vertical tab stop.
CTRL/L	Form feed.
CTRL/O	Alternately suppresses and continues display of output to the terminal.
CTRL/Q	Restarts terminal output that was suspended by CTRL/S.
CTRL/R	Retypes the current input line and leaves the cursor positioned at the end of the line.
CTRL/S	Suspends terminal output until CTRL/Q is pressed.
CTRL/U	Discards the current input line.
CTRL/X	Discards the current line and deletes data in the type-ahead buffer.
CTRL/Y	Interrupts command or program execution and returns control to the command interpreter.
CTRL/Z	Signals end-of-file for data entered from the terminal.
DELETE	Deletes the last character entered at the terminal and backspaces over it. (On some terminals, the DELETE key is labeled RUBOUT.)
ESCAPE	Has special uses to particular commands or programs, but generally performs the same function as RETURN. (On some terminals, the ESCAPE key is labeled ALTMODE.)
TAB	Moves the printing element or cursor on the terminal to the next tab stop on the terminal. The system provides tab stops at every 8 character positions on a line.

[1] Certain system and user programs provide special routines to respond to CTRL/C interrupts. If CTRL/C is pressed to interrupt a program that does not handle CTRL/C, CTRL/C has the same effect as CTRL/Y and echoes as ^Y.

1.10 SUMMARY OF VAX/VMS DCL COMMANDS

Tables 1-2 through 1-6 summarize the VAX/VMS DCL commands. The commands are grouped by functional category, as follows:

- Commands for terminal communication and control
- Commands for file manipulation
- Commands for device handling
- Commands for program development and control
- Commands for command procedures and batch jobs

Note that several commands are listed in more than one table and, in some cases, listed with a qualifier that is required to request a particular function.

Table 1-2
Commands for Terminal Communication and Control

Command	Function
:=	String assignment; defines a local symbol name as a synonym for all or a portion of a DCL command
:==	String assignment; defines a global symbol name as a synonym for all or a portion of a DCL command
DELETE/SYMBOL	Deletes a local or global symbol definition
HELP	Displays information about a command or a command parameter or qualifier on the terminal
LOGOUT	Terminates communication between a user and the system
MCR	Passes a command line to the RSX-11M Application Migration Executive, or places the terminal in MCR command mode
REQUEST	Displays a message at an operator's terminal
SET CONTROL_Y	Enables the use of the CTRL/Y function key to interrupt an image
SET NOCONTROL_Y	Disables the use of the CTRL/Y function key to interrupt an image
SET TERMINAL	Defines the characteristics of the terminal for the duration of a terminal session
SHOW DAYTIME	Displays the current date and time of day on the terminal
SHOW SYMBOL	Displays current local or global symbols and the strings or values assigned to them
SHOW TERMINAL	Displays the current characteristics of the terminal

Table 1-3
Commands for File Manipulation

Command	Function
APPEND	Adds the contents of one or more files to the end of another file
COPY	Copies one or more files into one or more additional files
CREATE	Creates a file from data entered at the terminal or in the input stream
CREATE/DIRECTORY	Defines a new directory or subdirectory for cataloging files
DELETE	Removes a directory entry for a file or files and makes any data in the file(s) inaccessible
DELETE/ENTRY	Deletes an entry from a printer or batch job queue or stops processing of the current job
DIFFERENCES	Compares the contents of files and reports the differences between them
DIRECTORY	Displays information about a file or a group of files
DUMP	Displays data in ASCII, hexadecimal, octal, or decimal format
EDIT	Begins an interactive editing session to create or modify a file
EDIT/SLP	Provides input to the batch editor, SLP
PRINT	Queues a copy of a file for printing on a system printer
PURGE	Deletes all but the most recent version or versions of a specified file or files
RENAME	Changes the name of a file or a group of files
SET DEFAULT	Changes the default directory and/or disk device used to locate and catalog files
SET PROTECTION	Changes the protection applied to a file or a group of files, restricting or allowing access to the file by different categories of user
SET QUEUE/ENTRY	Changes the attributes or status of an entry in the printer queue
SET RMS_DEFAULT	Defines default multi-block and multi-buffer counts for RMS file operations
SHOW DEFAULT	Displays the current default directory and disk device

(continued on next page)

Table 1-3 (Cont.)
Commands for File Manipulation

Command	Function
SHOW PRINTER	Displays characteristics of a line printer
SHOW PROTECTION	Displays the default protection applied to new files created
SHOW QUEUE	Displays the names and identifications of current and pending jobs in the printer and batch job queues
SHOW RMS_DEFAULT	Displays the current multi-block and multi-buffer counts for RMS operations
SORT/RSX11	Creates a sorted copy of a file, with records arranged in a particular collating sequence
TYPE	Displays the contents of a file or files at the terminal
UNLOCK	Allows access to a file that was not properly closed

Table 1-4
Commands for Device Handling

Command	Function
ALLOCATE	Reserves a device for use by a single user and, optionally, assigns a logical name to the device
ASSIGN	Defines a file specification or a device name to be associated with a logical name for subsequent use in commands or programs
DEALLOCATE	Relinquishes use of a previously allocated device, thus making the device available to other users
DEASSIGN	Cancels a logical name assignment made with the ALLOCATE, ASSIGN, DEFINE, or MOUNT commands
DISMOUNT	Releases the connection between a user and a disk or tape volume that is currently mounted on a device
INITIALIZE	Deletes all existing data, if any, on a mass storage volume, writes a label on the volume, and readies the volume for new data
MOUNT	Makes a disk or tape volume available for the reading and writing of files and, optionally, assigns a logical name to the device on which the volume is mounted

(continued on next page)

Table 1-4 (Cont.)
Commands for Device Handling

Command	Function
SET MAGTAPE	Defines the density of a magnetic tape device or rewinds a tape
SHOW DEVICES	Displays the status of devices in the system
SHOW LOGICAL	Displays current logical name assignments for a particular logical name or for all logical names
SHOW MAGTAPE	Displays characteristics of a magnetic tape device
SHOW TRANSLATION	Searches all three logical name tables for a logical name and displays the equivalence name of the first match found

Table 1-5
Commands for Program Development and Control

Command	Function
ASSIGN	Defines a file specification to be associated with a specific logical name used in a program
BASIC	Invokes the PDP-11 BASIC-PLUS-2/VAX compiler to enter and compile BASIC language source statements
CANCEL	Halts periodic execution of an image scheduled for execution in a process
COBOL/RSX11	Invokes the PDP-11 COBOL-74/VAX compiler to compile a set of COBOL language source statements
CONTINUE	Resumes execution of an interrupted command, program, or command procedure
DEASSIGN	Cancels a logical name assignment
DEBUG	Invokes the VAX-11 Symbolic Debugger to begin or continue interactive debugging
DEFINE	Equates character strings with file specifications or logical names
DEPOSIT	Replaces the contents of a location in virtual memory with new data or instructions
EDIT	Invokes an editor to create or modify a source program or data file

(continued on next page)

Table 1-5 (Cont.)
Commands for Program Development and Control

Command	Function
EXAMINE	Displays the contents of a location in virtual memory
FORTRAN	Invokes the VAX-11 FORTRAN IV-PLUS compiler to compile a set of FORTRAN language source statements
LIBRARY	Creates or modifies a macro library or a library of object modules
LINK	Binds one or more object modules into an executable or shareable program image
LINK/RSX11	Invokes the RSX-11M Task Builder to link one or more object modules into an executable image
MACRO	Invokes the VAX-11 MACRO assembler to assemble a VAX-11 assembly language program
MACRO/RSX11	Invokes the MACRO-11 assembler to assemble a PDP-11 assembly language program
RUN (Image)	Places an executable image in execution in the current process
RUN (Process)	Creates a separate process to execute a specified image
SET PROCESS	Defines execution characteristics of a process
SET WORKING_SET	Establishes a default working size for images executed in the current process
SHOW LOGICAL	Displays the current assignments of logical names and equivalence names made by the ASSIGN, ALLOCATE, DEFINE, or MOUNT commands
SHOW PROCESS	Displays information about the current process, including subprocesses, privileges, quotas, and accounting information
SHOW STATUS	Displays information about the image currently executing in the process
SHOW SYSTEM	Displays the current status of all processes in the system
SHOW TRANSLATION	Displays the result of logical name translation of a specific logical name
SHOW WORKING_SET	Displays the current working set default and limits
STOP	Halts execution of a command procedure, program, or a subprocess

Table 1-6
Commands for Command Procedures and Batch Jobs

Command	Function
@file-spec	Execute procedure; executes a command procedure or places data in a command file in the input stream
=	Arithmetic assignment; equates a local symbol name to an arithmetic expression or constant
==	Arithmetic assignment; equates a global symbol name to an arithmetic expression or constant
:=	String assignment; equates a local symbol name to any character string
:==	String assignment; equates a global symbol name to any character string
label:	Defines a statement label
CLOSE	Cancels an input or output path to a sequential file or device
CONTINUE	Requests continuation of the current procedure
DECK	Marks the beginning of records to be read as the input data stream for a command (required only when data contains a dollar sign in the first position of any record)
DELETE/ENTRY	Deletes a job from a printer or batch job queue
DELETE/SYMBOL	Deletes one or more symbol names from the local or global symbol tables for the process
EOD	Marks the end of an input data stream begun with the DECK command
EOJ	Signals the end of a batch job submitted through a system card reader
EXIT	Terminates command procedure processing at the current level
GOTO	Transfers control to another statement in a command procedure
IF ... THEN	Compares expressions consisting of symbolic or literal values or command or program status values and performs a stated action based on the result of the test
INQUIRE	Requests interactive assignment of a variable value for a symbol name

(continued on next page)

Table 1-6 (Cont.)
Commands for Command Procedures and Batch Jobs

Command	Function
JOB	Marks the beginning of a batch job submitted through a system card reader
ON ... THEN	Defines the action to be taken when a command or program incurs errors of particular severity levels, or when the CTRL/Y function key is used
OPEN	Establishes a path to a file or a device for input or output operations
PASSWORD	Provides a password associated with a job entered through a system card reader
READ	Reads the next record from a sequential file or device and equates the record data to a symbol name
SET CARD_READER	Defines the translation mode for a card reader
SET NOON	Suppresses command interpreter error checking following command execution
SET NOVERIFY	Suppresses display of command lines executed in command procedures subsequently executed
SET ON	Restores command interpreter error checking in a command procedure
SET QUEUE/ENTRY	Changes the attributes of a queued batch job
SET VERIFY	Causes all command lines in command procedures subsequently executed to be displayed at the terminal or printed in the batch job log file
STOP	Terminates command procedure processing at any level and returns control to the command interpreter
SUBMIT	Queues a command procedure to a batch job queue
SYNCHRONIZE	Places the current command procedure in wait state until a specified batch job completes
WAIT	Places the current process in a wait state for a specified period of time
WRITE	Writes a single record consisting of one or more character strings or evaluated symbols to a sequential file or device

Table 1-7
User Privileges

Name	Privilege
ACNT	Create a process for which no accounting records are made
ALLSPOOL	Allocate spooled devices
ALTPRI	Increase the base execution priority for any process
BUGCHK	Make bug check error log entries
CMEXEC	Change mode to executive
CMKRNL	Change mode to kernel
DETACH	Create detached processes
DIAGNOSE	Issue diagnostic I/O requests
EXQUOTA	Exceed resource quotas
GROUP	Control execution of other processes in the same group
GRPNAM	Enter names in the group logical name table
LOG_IO	Perform logical I/O functions
MOUNT	Execute a mount volume I/O function
NETMBX	Create a network device
OPER	Perform operator functions
PHY_IO	Perform physical I/O functions
PRMCEB	Create permanent common event flag clusters
PRMGBL	Create permanent global clusters
PRMMBX	Create permanent mailboxes
PSWAPM	Disable swapping for any process
SETPRV	Grant a created process any privileges
SYSGBL	Create system global sections
SYSNAM	Enter names in the system logical name table
TMPBMX	Create temporary mailboxes
VOLPRO	Override protection on a volume
WORLD	Control the execution of any process in the system

Table 1-8
Resource Quotas

Name	Quota
ASTLM	AST (Asynchronous System Trap) limit
BIOLM	Buffered I/O limit
BYTLM	Buffered I/O byte count (buffer space) quota
CPULM	CPU time limit
DIOLM	Direct I/O limit
FILLM	Open file quota
PGFLQUOTA	Paging file quota
PRCLM	Subprocess quota
TQELM	Timer queue entry quota
WSDEFAULT	Default working set size
WSQUOTA	Working set size quota

CHAPTER 2

FILE SPECIFICATIONS AND LOGICAL NAMES

A file is a logically related collection of records. All the information that the operating system reads and writes on behalf of users' requests is defined in terms of files and records.

Files are identified by the hardware device that performs the actual data transfer (reading or writing). Devices are classified as:

- Mass storage devices
- Record-oriented devices

Mass storage devices provide a way to save the contents of files on a magnetic medium, called a volume. Files that are thus saved can be accessed at any time and updated, modified, or reused. Disks and tapes are mass storage devices.

Record-oriented devices read and write only single physical units of data at a time, and do not provide for permanent storage of the data. Terminals, printers, and card readers are record-oriented devices. Printers and card readers are also called unit record devices.

This chapter discusses:

- How to specify devices and files when you enter DCL commands
- How to construct and use logical names to refer to files and devices

You can find additional specific information about how to use DCL commands to manipulate files in the command descriptions in Part II. Table 1-3 lists the commands that provide file handling capabilities. Chapter 3, "Disk and Tape Volumes" provides more information on how to handle files on mass storage devices.

2.1 FILE SPECIFICATIONS

File specifications provide the system with all the information it needs to identify a unique file or device.

File specifications have the format:

```
node::device:[directory]filename.type;ver
```

The punctuation marks and brackets are required to separate the fields of the file specification. The fields are:

Field	Contents
node	Network node name
device	Device name
directory	Directory name
filename	File name
type	File type
ver	File version number

File names, file types, and version numbers apply only to files on mass storage devices. For record-oriented devices, only the device name field in the file specification is required.

Additional notes and syntax requirements for each field in a file specification are discussed below. Note that you do not have to enter a complete file specification each time you specify a file; the system supplies defaults for unspecified fields. Section 2.1.5, "Defaults for File Specifications" describes defaults in more detail.

2.1.1 Network Nodes

A node name is a 1- to 6-alphanumeric character name that identifies a location on the network, if your system is connected to one. If you specify a node name, you can optionally include a 1- to 30-character access control string, in the format:

```
node"access-control-string"::
```

Enclose the remainder of the file specification in quotation marks. The file specification is passed intact to the network node specified and is interpreted there. The portion on the right of the double colon (::) can have up to 125 characters.

For details on the syntax of file specifications and information on using DCL commands for network operations, see the DECnet-VAX User's Guide.

2.1.2 Devices

Each physical hardware device in the system is uniquely identified by a device name specification in the format:

```
devcu:
```

where dev is a mnemonic for the device type, c is a controller designation and u is a unit number.

Table 2-1 lists the valid device types and their mnemonics.

The controller and unit number identify the location of the actual device within the hardware configuration of the system. Controllers are designated with alphabetic letters A through Z. Unit numbers are decimal numbers from 0 through 65535.

The maximum length of the device name field, including controller and unit number, is 15 characters. When you specify a device name, terminate it with a colon (:).

Table 2-1
Device Names

Mnemonic	Device Type
CR	Card Reader
DB	RP04, RP05, RP06 Disk
DM	RK06, RK07 Disk
DR	RM03 Disk
DX	Floppy Disk
LP	Line Printer
MB	Mailbox
MT	TE16 Magnetic Tape
NET	Network communication device
TT	Interactive terminal
XM	DMC-11

A complete device name specification is called a physical device name. You can specify physical device names to indicate an input or output device for a command or program. Or, you can equate a physical device name to a logical name and use a logical name to refer to a device. Logical names are described in detail in Section 2.2.

Some commands allow you to specify a generic device name. A generic device name is one in which the controller and/or the unit number are not specified. When you use a generic device name, the system locates an available device whose physical name satisfies the portions of the generic device name that are specified. For example, if you issue an ALLOCATE command and specify only a device type, the ALLOCATE command locates an available device of that type.

2.1.3 Directories

A directory is a file that lists the identifications and locations of files on a disk device that are owned by a particular user. Directory names apply only to files on disk devices. Directory names have three formats:

- A 1- to 9-alphanumeric character string
- A two-part number in the format of a user identification code (UIC)
- As subdirectories, in the format of name.name.name where each name can consist of up to 9 alphanumeric characters; each name represents a directory level.

All these formats require the directory name to be enclosed in either square brackets ([and]), or in angle brackets (< and >).

2.1.3.1 **Directory Names** - An alphanumeric directory name can be the same as your user name or account name, or any character string that you request or that the system manager gives you. For example:

```
[MALCOLM]
```

To specify a directory name in UIC format, separate the group number from the member number with a comma. For example:

```
[122,1]
```

Directories in UIC format generally, but not necessarily, correspond to the UIC of the owner of the directory.

UIC directories can be expressed in alphanumeric format. The group and member numbers are each left zero-filled on the left (if necessary). For example:

```
[122001]
```

This directory specification is equivalent to the specification [122,1] in the preceding example.

To specify a subdirectory, separate directory level identifiers with periods. For example:

```
[MALCOLM.TESTFILES]
[122001.TESTFILES.DATA]
```

You cannot specify a directory name in UIC format if the directory includes a subdirectory, for example [122,1.SUB] is invalid. If you have a UIC directory, you must specify it in alphanumeric format when you are specifying subdirectories.

2.1.3.2 **Directory Hierarchies** - You must have at least one directory, provided by the system manager, before you can create and catalog files on system disks. Optionally, you can create, in your own directory, one or more directory level hierarchies.

The CREATE command can create a subdirectory. For example, the following command creates a second-level directory for the directory named MALCOLM:

```
$ CREATE/DIRECTORY [MALCOLM.SUB]
```

This command places an entry for the directory file SUB.DIR in the first-level directory MALCOLM. Subsequently, you can use the subdirectory name [MALCOLM.SUB] in a file specification.

A subdirectory can contain an entry for another directory; that directory can contain an entry for another directory, and so on. The maximum number of levels, including the first level directory, is eight. This structure constitutes a directory hierarchy. Figure 2-1 illustrates directory hierarchies.

There is no maximum on the number of hierarchies of directories you can create and access beginning with your own directories.

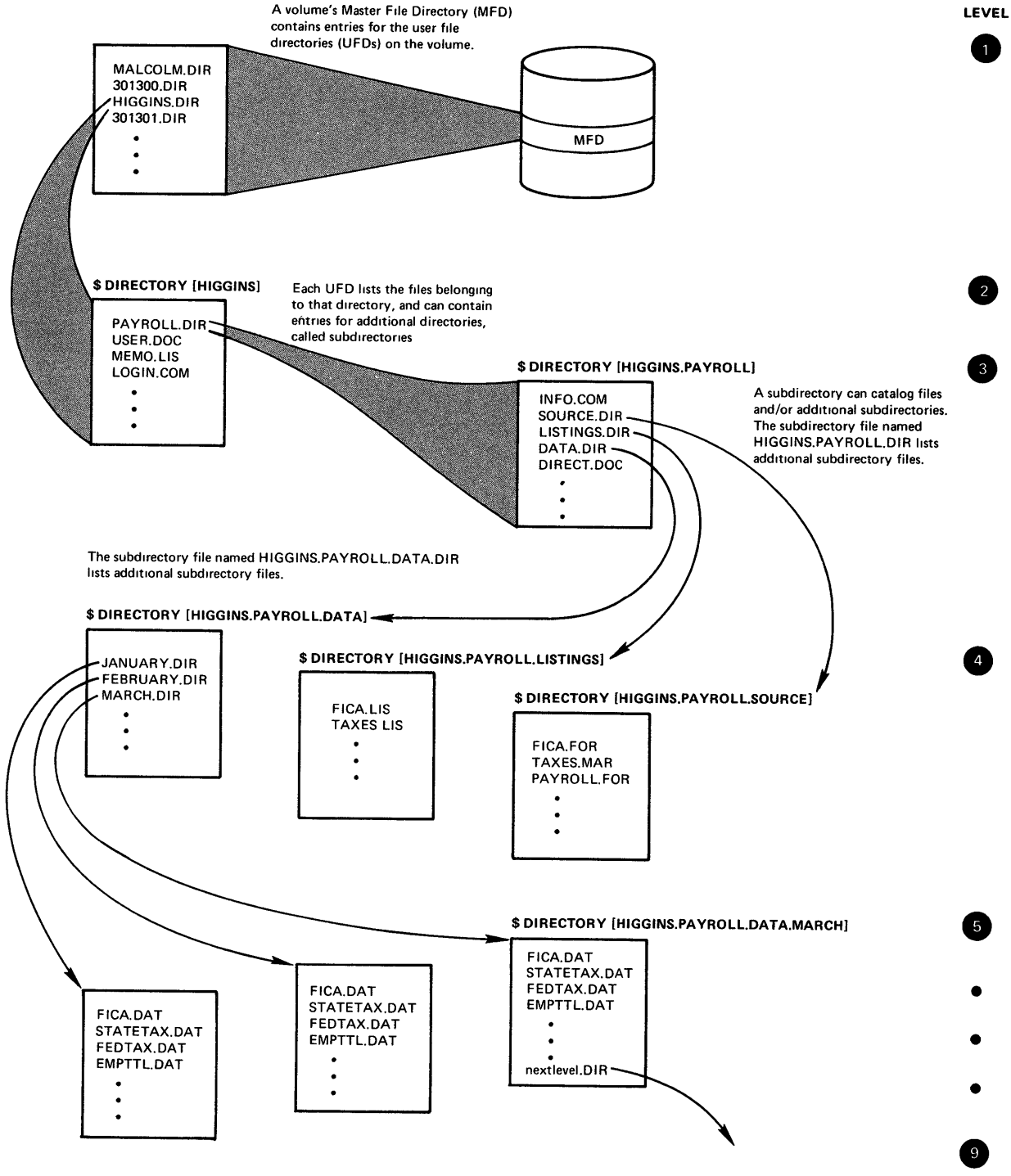


Figure 2-1 A Directory Hierarchy

2.1.4 File Names, File Types, and Version Numbers

File names, file types, and version numbers uniquely identify files within directories.

A file name is a 1- to 9-character string name for a file. When you create a file, you can assign it a file name that is meaningful to you.

A file type is a 1- to 3-character string that usually identifies the file in terms of its contents.

The valid characters in file names and file types are A-Z and 0-9.

File types must be preceded with a period (.).

By convention, VAX/VMS uses a set of standard file types to identify various classifications of files, and to provide default file types in many commands. Table 2-2 lists these file types.

Version numbers are decimal numbers from 1 to 32767 that differentiate between versions of a file. When you update or modify a file and do not specify a version number for the output file, the system saves the original version for backup and increments the version number by 1. Note, however, that on Files-11 Structure Level 2 disks, the file system deletes the lowest numbered versions of a file after more than approximately 60 versions of the file exist.

Version numbers must be preceded with a semicolon (;) or a period (.).[1]

Table 2-2
Default File Types

File Type	Contents
B2S	Input source file for the PDP-11 BASIC-PLUS-2/VAX compiler
CMD	Compatibility mode indirect command file
COM	Command procedure file to be executed with the @ (Execute Procedure) command, or to be submitted for batch execution with the SUBMIT command
CBL	Input file containing source statements for the PDP-11 COBOL-74/VAX compiler
DAT	Input or output data file
DIF	Output listing created by the DIFFERENCES command
DIR	Directory file
DMP	Output listing created by the DUMP command

(continued on next page)

[1] When the system displays file specifications, it generally displays a semicolon in front of the file version number.

Table 2-2 (Cont.)
Default File Types

File Type	Contents
EXE	Executable program image
FOR	Input file containing source statements for the VAX-11 FORTRAN-IV-PLUS compiler
LIS	Listing file created by a language compiler or assembler; default input file type for PRINT and TYPE commands
LOG	Batch job output file
LST	Compatibility mode listing file
MAC	MACRO-11 source file
MAP	Memory allocation map created by the linker
MAR	VAX-11 MACRO source file
MLB	Macro library
OBJ	Object file created by a language compiler or assembler
ODL	Overlay description file
OLB	Object module library
OPT	Options file for input to the LINK command
STB	Symbol table file created by the linker

2.1.5 Defaults for File Specifications

When you enter a file specification, you can omit fields in the specification and let the system supply values for these fields. The values supplied by the system are called defaults.

The device and directory names, if omitted, default to your current default disk and directory name. These are initially established when you log in, based on an entry under your user name in the system authorization file.

You can determine your default disk and directory name by issuing the SHOW DEFAULT command. For example:

```
$ SHOW DEFAULT
  DBA1:[RABBIT]
```

This response indicates that the current default disk is DBA1 and the directory name is RABBIT.

You can change the disk and directory defaults during a terminal session or in a batch job by using the SET DEFAULT command. For example, if the default disk and directory are as shown in the response to the SHOW DEFAULT command above, the following SET DEFAULT command would change the default directory to a subdirectory:

```
$ SET DEFAULT [RABBIT.SUB]
```

The device and directory are not the only fields for which the system supplies default values. Table 2-3 summarizes the defaults, if any, applied to each field in a file specification.

Table 2-3
File Specification Defaults

Field	Defaults
node	Local system.
device	Device (usually a disk) established at log in, or by the SET DEFAULT command. If a controller designation is omitted, it defaults to A. If a unit number is omitted, it defaults to 0. (The ALLOCATE and SHOW DEVICE commands, however, treat a device name that does not contain controller and/or unit numbers as a generic device name. For more details, see the discussions of these commands in Part II.)
directory	Directory name established at log in or by the SET DEFAULT command. If a first- or sub-level directory name is omitted, it defaults to the current default next higher directory names, as long as the default directory name string was specified in alphanumeric format. The subdirectory name must be preceded with a period, for example [.SUB].
file name	No defaults are applied to the first file name in an input file specification. Most commands apply default output file names based on the file name of an input file.
file type	Various commands apply defaults for file types, based on the standard file type conventions summarized in Table 2-2.
file version	For input files, the system assumes the highest version number. For output files, the system increases the highest existing version number by 1 for existing files, and supplies a version number of 1 for new files.

2.1.5.1 **Temporary Defaults** - All DCL commands that accept lists of input files apply temporary defaults when you enter a command line that contains more than one input file specification. Temporary defaulting lets you name a device, directory, file name, or file type that all the files you specify have in common. The system uses temporary file specification defaults only to interpret file specifications for a single execution of a command. Temporary defaults are applied to:

- Node name
- Device name
- Directory name
- File name and file type

If a file specification explicitly includes a device and/or directory name, these specifications become the temporary defaults for the interpretation of subsequent file specifications within the list.

File names and file types can also be defaulted, depending on the specific command.

For example, assume that the current default disk device and directory name are DBB2:[MONROE]. The following PRINT command shows how temporary defaults are applied to a list of file specifications in a parameter:

```
$ PRINT DBA1:[ADAMS]TEST1.DAT, -
$_  TEST2, -
$_  [JACKSON]SUMMARY.TST, -
$_  DBB2:FINAL
```

The PRINT command prints the files:

```
DBA1:[ADAMS]TEST1.DAT
DBA1:[ADAMS]TEST2.DAT
DBA1:[JACKSON]SUMMARY.TST
DBB2:[JACKSON]FINAL.TST
```

To override a temporary default to specify your current default directory, you can specify the directory as shown below:

```
$ PRINT  [ALPHA]TEST.DAT,[]TEST
```

Brackets with no directory name indicate the system is to use your current default directory.

2.1.5.2 **Null File Names and File Types** - The file name and file type fields of a file specification can be null. For example, the following are valid file specifications:

```
.TMP
TEMP.
```

When you specify a file in a DCL command, you must be careful to omit the period following a file name if the command uses a default file type. For example, the FORTRAN command uses a default file type of FOR. The following commands produce different results:

```
$ FORTRAN TEMP
$ FORTRAN TEMP.
```

In the first example, compiler looks for a file named TEMP.FOR because the file type was omitted. In the second example, the compiler looks for a file named "TEMP." because a period following the file name indicates a null file type.

2.1.6 Wild Cards

Many DCL commands accept a special character, called a wild card, in file specifications. The asterisk (*) represents a wild card. Wild cards are valid in the directory, file name, file type, or file version fields. The precise meaning of the wild card depends on whether the file specification is for an input or an output file.

2.1.6.1 **Wild Cards for Input Files** - When a wild card replaces a field in an input file specification, the system locates all files whose identifications satisfy the fields that are specified.

The DIRECTORY command accepts the file specification of an input file and displays information about the file at the terminal. The following examples of the DIRECTORY command illustrate how DCL commands interpret wild cards in input file specifications.

Example		Explanation
DIRECTORY	DBB2:[*]FILES.DAT	Displays the highest version of the file FILES.DAT in all first level (user file) directories on the device DBB2
DIRECTORY	*.LIS	Displays the highest version of each file with a file type of LIS from the current default disk and directory
DIRECTORY	TEST.*	Displays the highest version of each file from the current default disk and directory that has a file name of TEST
DIRECTORY	TEST.FOR;*	Displays all versions of the file TEST.FOR from the current default disk and directory

2.1.6.2 **Wild Cards for Output Files** - When a wild card replaces a field in an output file specification, the system uses the corresponding field in the input file specification to fill in the wild card field of the output file.

The COPY command copies the contents of a specified input file into a new output file. The following examples of the COPY command illustrate how DCL commands interpret wild cards in output file specifications.

Example	Explanation
COPY TEST.DAT *.OLD	Copies the highest version of the file TEST.DAT from the current default disk and directory into a file named TEST.OLD
COPY [CHEVY]*.FOR *	Copies the highest version of each file with a file type of FOR in the directory CHEVY on the current default disk to new files in the current default directory

Particular uses of wild cards in DCL commands vary with the individual commands. The command descriptions in Part II of this manual describe wild usage where applicable. Note that not all DCL commands accept wild cards; moreover, wild cards are always invalid in files specified for network operations, and within the specification of any portion of a directory name string that contains a subdirectory name.

2.2 LOGICAL NAMES

Logical names allow you to keep programs and command procedures independent of physical file specifications. They also provide a convenient shorthand way to specify files that you refer to frequently.

The ASSIGN command equates a physical file specification to a logical name, that is, to a character string name that you supply. For example:

```
$ ASSIGN DBA2:[SIMMONS]MARIGOLD.DAT  TEST
```

This ASSIGN command equates the logical name TEST to the file specification DBA2:[SIMMONS]MARIGOLD.DAT. This file specification is called the equivalence name for the logical name. Subsequently, you can refer to this file by its logical name when you type a DCL command. For example:

```
$ TYPE TEST
```

When the system processes this command, it replaces the logical name TEST with its equivalence name, and displays the contents of the file MARIGOLD.DAT on the terminal.

You can also assign logical names to devices when you issue ALLOCATE, DEFINE, or MOUNT commands.

2.2.1 Logical Name Tables

The system maintains logical name and equivalence name pairs in three logical name tables:

- Process logical name table -- contains logical name entries that are local to a particular process. By default, the ASSIGN command places a logical name in the process logical name table.

- Group logical name table -- contains logical name entries that are qualified by a group number. These entries can be accessed only by processes that execute with the same group number in their user identification codes as the process that assigned the logical name. You must use the /GROUP qualifier to make an entry in the group logical name table.

- System logical name table -- contains entries that can be accessed by any process in the system. You must use the /SYSTEM qualifier to make an entry in the system logical name table.

The user privileges GRPNAM and SYSNAM are required to place entries in the group or system logical name tables, respectively.

2.2.2 How to Specify Logical Names

Logical names and their equivalence name strings can each have a maximum of 63 characters, and can be used to form all or part of a file specification. If only part of a file specification is a logical name, it must be the left-most component of the file specification. You can then specify the logical name in place of the device name in subsequent file specifications, terminated by a colon (:).

For example, a logical name can be assigned to a device name, as follows:

```
$ ASSIGN DMA1: BACKUP:
```

After this ASSIGN command, you can use the logical name BACKUP in place of the device name field when referring to the device. For example, the following COPY command transfers files from the current default disk and directory to a directory with the same name on the volume on the device DMA1 by using the logical name BACKUP.

```
$ COPY *.* BACKUP:
```

A logical name can also contain both a device name and directory name. For example:

```
$ ASSIGN DMA1:[MAGGIE] SCRATCH
```

This ASSIGN command assigns the logical name SCRATCH to the directory named MAGGIE on the device DMA1. You can now use the logical name SCRATCH in a file specification, as in the example below.

```
$ PRINT SCRATCH:PAYROLL.DAT
```

The PRINT command prints the file PAYROLL.DAT from the directory [MAGGIE] on DMA1.

When you specify an equivalence name for the ASSIGN command, you must specify it using the proper punctuation marks (colons, brackets, periods). If you specify only a device name, terminate the equivalence name parameter with a colon (:); if you specify a device and directory name, or a full file specification, do not terminate the equivalence name with a colon.

You can optionally terminate the logical name parameter with a colon; however, the ASSIGN command removes the colon before placing the name in the logical name table.[1]

2.2.2.1 **Displaying Logical Name Table Entries** - The SHOW LOGICAL command displays current entries in the logical name tables. For example, to display the logical name SCRATCH, you would enter the command:

```
$ SHOW LOGICAL SCRATCH
  SCRATCH  =  DMA1:[MAGGIE]      (process)
```

The SHOW LOGICAL command displays the logical name, its equivalence name, and the logical name table in which it found the name.

You can also request the system to display all the entries in a specified logical name table. For example:

```
$ SHOW LOGICAL/GROUP
```

This SHOW LOGICAL command results in a display of all current entries in the group logical name table.

2.2.3 Logical Name Translation

When the system reads a device name or a file specification, it examines the file specification to see if the left-most component is a logical name. If it is, the system substitutes the equivalence name in the file specification. This is called logical name translation.

For example:

```
$ TYPE ALPHA
$ TYPE DISK:ALPHA
```

When the system reads the file specification ALPHA in the first example, it checks to see if ALPHA is a logical name because ALPHA is the left-most (and in this example, the only) component of the file specification. In the second example, the system checks to see if DISK is a logical name because it is the left-most component. It does not check ALPHA.

When the system translates logical names, it searches the process, group, and system tables, in that order, and uses the first match it finds.

If you are ever in doubt about the current equivalence name assigned to a logical name, use the SHOW LOGICAL command.

---

[1] The DEFINE command, which also creates logical name table entries, does not remove colons, if specified, from logical names. It is recommended that you do not use the DEFINE command for device name assignments.

2.2.3.1 **Recursive Translation** - When the system translates logical names in file specifications, the logical name translation is often recursive. This means that after the system translates a logical name in a file specification, it repeats the process of translating the file specification. For example, consider logical name table entries made with ASSIGN commands as follows:

```
$ ASSIGN DBA1:  DISK
$ ASSIGN DISK:WEATHER.SUM REPORT
```

The first ASSIGN command equates the device DBA1 to the logical name DISK. The second ASSIGN command equates the logical name REPORT to the file specification DISK:WEATHER.SUM. In subsequent commands, or in programs you execute, you can refer to the logical name REPORT. For example:

```
$ TYPE REPORT
```

When the system translates the logical name REPORT, if finds the equivalence name DISK:WEATHER.SUM. It then checks to see if the portion to the left of the colon in this file specification is a logical name; if it is (as DISK is in this example), it translates that logical name also. When the logical name translation is complete, the translated file specification is:

```
DBA1:WEATHER.SUM
```

Note that when you assign one logical name to another logical name, you must terminate the equivalence name with a colon (:) if you are going to use the logical name in a file specification in place of a device name. For example:

```
$ ASSIGN DBA1:  TEST
$ ASSIGN TEST:  GO
$ TYPE GO:NEW.DAT
```

The TYPE command types the file NEW.DAT from the disk DBA1. If you omit the colons from either of these ASSIGN commands, the system will not be able to form a proper file specification.

2.2.3.2 **Applying Defaults** - When the system completes the translation of a logical name, it uses defaults to fill in any still unspecified fields in the file specification. In the above examples, the system completes the file specifications by supplying the current default directory and a version number according to whether it is an input or an output file.

Many system commands create output files automatically and provide default file types for the output files. When you use a logical name to specify the input file for a command, the command uses the logical name to assign a file specification to the output file as well. Thus if the equivalence name contains a file name and file type, the output file is given the same file name and file type as the input file but a higher version number.

For example, the LINK command creates, by default, an executable image file that has the same file name as the input file and a default file type of EXE. If you make a logical name assignment and invoke the LINK command as shown below the results are not as you would expect:

```
$ ASSIGN RANDOM.OBJ TESTIT
$ LINK TESTIT
```

The linker translates the logical name TESTIT and links the file RANDOM.OBJ. When it creates the output file, it also uses the same logical name for the output file. Because the equivalence name has a file type, the LINK command does not use the default file type of EXE. In this example, the executable image is named RANDOM.OBJ and it has a version number one higher than the version number of the input file.

2.2.3.3 **Logical Names in Input File Lists** - When you issue a command that accepts multiple input files and you use logical names in the specifications of one or more files in the list, the equivalence name of each logical name provides a temporary default. For example:

```
$ SET DEFAULT  DBA2:[CASEY]
$ ASSIGN  DBA1:[MALCOLM] MAL
$ ASSIGN  [HIGGINS] HIG
$ PRINT  ALPHA,-
$_ MAL:BETA,-
$_ HIG:GAMMA
```

The PRINT command looks for the files:

```
DBA2:[CASEY]ALPHA.LIS
DBA1:[MALCOLM]BETA.LIS
DBA1:[HIGGINS]GAMMA.LIS
```

The device name in the equivalence string for the logical name MAL defines DBA1 as the temporary default device for this PRINT command. Temporary defaults are decribed in Section 2.1.5.1. For complete details on file specification defaults, see the VAX-11 Record Management Services Reference Manual.

2.2.3.4 **Bypassing Logical Name Translation** - Most DCL commands check file specifications you enter as command parameters or as values for qualifiers to see if the file specification contains a logical name. When you enter a device name in a command, you can request the system to not translate the name by preceding the device name or file specification with an underscore character (_). For example:

```
$ ALLOCATE _DMA2:
```

When you specify an ALLOCATE command as shown, the system does not check to see if DMA2 is a logical name.

2.2.4 Default Logical Names

The operating system and many of its facilities use logical names to establish default devices for input and output operations. By convention, logical names defined for VAX/VMS functions have the format:

```
xxx$name
```

where xxx is a three-character prefix identifying the system component that uses the logical name.

2.2.4.1 **Default Process Logical Names** - When you log in to the system, the system creates logical name table entries for your process. These logical names, which all have a prefix of SYS, are listed in Table 2-4.

Table 2-4
Default Process Logical Names

Logical Name	Equivalence Name
SYS$INPUT	Default input stream for the process. For an interactive user, SYS$INPUT is equated to the terminal. In a command procedure or batch job, SYS$INPUT is equated to the input file or batch input stream.
SYS$OUTPUT	Default output stream for the process. For an interactive user, SYS$OUTPUT is equated to the terminal. In a batch job, SYS$OUTPUT is equated to the batch job log file.
SYS$ERROR	Default device to which the system writes messages. For an interactive user, SYS$ERROR is equated to the terminal. In a batch job, SYS$ERROR is equated to the batch job log file.
SYS$COMMAND	Original SYS$INPUT device for an interactive user or batch job.
SYS$DISK	Default device established at login, or changed by the SET DEFAULT command.

The equivalence names for SYS$INPUT, SYS$OUTPUT, SYS$ERROR and SYS$COMMAND define files that remain open for the life of the process. These files, called process-permanent files, can be read or written from programs.

2.2.4.2 **Using Default Logical Names** - The default logical name assignments are provided for your convenience, and do not normally need to be reassigned. The system always uses SYS$INPUT, SYS$OUTPUT, and SYS$ERROR to read and write commands, data, and messages.

You can take advantage of the default assignments when you specify files in commands. For example, to place input data in the command stream for a command or program, you can specify an input file as SYS$INPUT. The following example shows a FORTRAN command that could be executed in a batch job:

```
$ FORTRAN SYS$INPUT:WEATHER
```

When this command executes, the compiler reads the input file from the command stream; if this job is submitted through the system card reader, the cards containing the source program must follow in the card deck. The file name WEATHER in the file specification provides the compiler with a default file name for the output files: it creates WEATHER.OBJ and WEATHER.LIS.

To request that the listing file be written directly into the batch job output log file, you could specify:

```
$ FORTRAN/LIST=SYS$OUTPUT   SYS$INPUT:WEATHER
```

The compiler creates the file WEATHER.OBJ, but prints the listing in the batch job output log.

2.2.4.3 **Default System Logical Names** - The system logical name table contains entries for system-wide logical names. Among these logical names are:

SYS$LIBRARY	Device and directory name of system libraries
SYS$SYSTEM	Device and directory of operating system programs and procedures

The system manager at your installation can place names in the system logical name table that correspond to default devices on your particular system.

2.2.5 Logical Names for Program Input/Output

Logical names are also used to establish the correspondence between a file name or logical unit number in a program and a physical device or file specification. Each programming language provides conventions for identifying files within programs; the operating system provides the logical name mechanism for equating these files with physical device or file specifications.

For example, a FORTRAN program refers to a file using a logical unit number. FORTRAN run-time procedures associate the logical unit numbers with logical names: logical unit 1 is associated with the logical name FOR001, logical unit 2 is associated with the logical name FOR002, and so on. Before running a program that reads from or writes to logical unit 1, you can make an assignment of that logical unit to a device with the ASSIGN command, as shown in the following example:

```
$ ASSIGN  PAYROLL.DAT  FOR001
$ RUN FICA
```

Before you execute programs to read and write files, you must take steps to ensure that the data volumes or devices that your program requires are online and available. The next chapter discusses how to prepare and use disk and tape volumes.

CHAPTER 3

DISK AND TAPE VOLUMES

You can use your default disk directory to catalog the files that you use on a regular basis. The physical disk volume that contains your default directory contains the files belonging to other users as well; such volumes are known as system volumes. Under normal operating circumstances, you do not need to perform any special action to access your own files.

Occasionally, you may also need to access files belonging to other users, or to transfer files to or from volumes other than the system volume containing your default directory. In these cases, special action may be required.

This chapter describes:

- How the operating system protects and restricts access to files and devices
- Volume initialization
- Mounting volumes on devices
- The steps necessary to transfer files to and from disk and tape volumes
- Accessing files from batch jobs

3.1 PROTECTION

The operating system protects data on disk and tape volumes to ensure against accidental or unauthorized access. Protection is provided at two levels:

- At the volume and file level to ensure that data on a volume is protected
- At the device level to ensure that no other users can access a device in use by one user

3.1.1 Volume and File Protection

Disk and tape volumes, and individual files on disk volumes, including directories, are protected by means of a protection code. The protection code indicates who is allowed access for what purposes.

Four categories of user are defined according to UIC. These are:

- SYSTEM -- all users who have group numbers of 1 through 10 (octal). These group numbers are generally for system managers, system programmers, and operators. Users who have either of the user privileges LOG_IO and PHY_IO are also included in this category.
- OWNER -- the UIC of the person who created, and therefore owns, the volume or file
- GROUP -- all users who have the same group number in their UICs as the owner of the file, including the owner
- WORLD -- all users who do not fall into any of the other three categories but including all users in the other categories

Each of these categories of user can be allowed or denied one of the following types of access:

- READ -- the right to examine, print, or copy a file or files on a volume
- WRITE -- the right to modify the file or to write files on a volume
- EXECUTE -- the right to execute files that contain executable program images (when applying protection to an entire volume, this field is interpreted as the right to create files on the volume)
- DELETE -- the right to delete the file or files on the volume

The system provides a default protection code for files you create. When you create a file or prepare a disk or tape volume for private use, you can define the protection you want to be applied. Some examples of specifying protection codes for individual files are shown in the following section; examples of volume protection are shown in Section 3.4, "Using Disk and Tape Volumes." For details on how to specify protection codes, see the description of the SET PROTECTION command in Part II.

3.1.1.1 **Disk File Protection** - Each file on a disk has its own protection code; you can specify a protection code when you create a file. For example, you can use the /PROTECTION qualifier to define the protection for a file you create with the COPY command, as shown below:

```
$ COPY DBA1:[PAYDATA]PAYROLL.DAT  PAYSORT.DAT -
$_    /PROTECTION=(SYSTEM:RW,OWNER:RWED,GROUP:RW,WORLD)
```

This COPY command copies a file from the device DBA1: to your default disk directory. The protection code defines the protection for the newly created file PAYSORT.DAT as follows: users with system UICs can read and write the file; you (the owner) have all types of access; other users in your group may read and write the file; and all other users (the world) are permitted no access.

You can also change the protection for an existing file with the SET PROTECTION command. For example:

```
$ SET PROTECTION=(SYSTEM:RWE,OWNER:RWED,GROUP:RE,WORLD) -
$_ PAYSORT.EXE
```

If you do not define a protection code for a file when you create the file, the system applies a default protection. The default protection is determined from an entry in the user authorization file. You can determine the current protection that is applied by issuing the SHOW PROTECTION command:

```
$ SHOW PROTECTION
  SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=R
```

This response indicates that the system and owner have all types of access, that other users in the owner's group have read and execute access, and that all other users (the world) are permitted read access.

To determine the current protection associated with a specific file or files, use the /FULL qualifier on the DIRECTORY command. For example:

```
$ DIRECTORY/FULL  PERSONNEL.REC

DIRECTORY DBA1:[CRAMER]
5-JUL-1978 12:31

PERSONNEL.REC;5        (3725,12)     8./10  03-JUL-1978 13:14
 [320,20]    [RWED,RWED,RW,R]

TOTAL OF 8./10.  BLOCKS IN 1.  FILE
```

You can change the default protection applied to files that you create during a terminal session with the SET PROTECTION command. Use the /DEFAULT qualifier on the SET PROTECTION command to indicate that the protection code you specify is to be applied to all files that you subsequently create during the terminal session or batch job.

3.1.1.2 **Tape File Protection** - The protection applied to a tape volume applies equally to all files on the volume. The system only applies read and write access restrictions with respect to tapes; execute and delete access are meaningless. Moreover, the system and the owner are always given both read and write access, regardless of what you specify in a protection code. If you give write access to the group or world, read access is also allowed.

3.1.2 Device Allocation

In most installations, many users must share a limited number of private disk drives and tape drives. The operating system controls access to these devices so that while you are using a device, no other user can access the device and the volume mounted on it. This control mechanism is called device allocation.

The ALLOCATE command allocates a device and gives you exclusive use of it. For example, to use the tape drive labeled B1, you would issue the following ALLOCATE command:

```
$ ALLOCATE MTB1:
  _MTB1: ALLOCATED
```

The response from the ALLOCATE command indicates successful allocation of the device you requested. You can also use the SHOW DEVICES command to find out what devices are available. For example:

```
$ SHOW DEVICES  DM:
```

This command requests a display of all RK06 and RK07 devices attached to the system. The response from the command shows the devices that are currently allocated to other users, and devices that are online and available.

If you want to allocate any device of a particular type, use a generic device name in the ALLOCATE command. A generic device name is one that does not specify a controller and/or a unit number. For example, if you want to use an RK06 device and do not know which drives are available, you would issue the ALLOCATE command as follows:

```
$ ALLOCATE DM:
  _DMA2: ALLOCATED
```

The ALLOCATE command locates an available RK06 device; the response indicates that the device DMA2 is allocated.

Note that tape devices can never be shared; thus, when you use a tape, the system automatically allocates it for you when you mount it. You can, however, allocate a tape device when you want to mount more than one volume on the same device and retain control of the device between mounts.

3.2 VOLUME INITIALIZATION

Before a volume can be used to contain files, it must be initialized. The INITIALIZE command:

- Invalidates all existing data on the volume, if any, and formats the volume with a particular data structure
- Writes an internal label on the volume to identify it
- Defines the owner UIC and protection for the volume

If a disk or tape volume has never been used before, no special steps are needed to initialize it. However, if the volume has previously contained data, the protection code on it may prevent you from accessing the volume and initializing it. Or, in the case of a tape, files on the volume may not have reached their expiration dates.

If either of these conditions exists and you do not have the user privilege to override volume protection, the previous owner of the volume or another user (the system manager or operator, for example) who does have read/write access must initialize it for you. When you give the volume to another user for initialization, you should specify:

- The data format you require
- The label you want to have written on the volume
- The protection code and owner UIC you want assigned to it

When you obtain a tape or disk volume, you should also remember to place a label on the outside of the volume so that it can be easily identified.

3.3 MOUNTING VOLUMES ON DEVICES

Mounting is the mechanism that provides a link between a volume, a device, and your process so you can perform input/output operations. Mounting and using a physical volume (a disk pack or tape reel) involves two distinct operations:

- Place the volume on the device and ready the device by pressing the load button or performing equivalent startup procedures.
- Issue the MOUNT command to gain access to the data on the device. The MOUNT command compares your UIC with the UIC of the volume's owner to verify your right to access the volume; it also verifies the label on the volume against the label you specify.

The order in which these operations are performed varies according to the way in which the device is being used. If you have already allocated a device using an explicit ALLOCATE command, you can physically load the device before issuing the MOUNT command. For example:

```
$ MOUNT DMA2: TEST_FILES INFILE
%MOUNT-I-MOUNTED, TEST_FILES mounted on _DMA2:
```

In this example, the MOUNT command specifies the name of an allocated device (DMA2), the volume label on the volume (TEST_FILES), and a logical name for the device (INFILE). The logical name parameter is optional. Note that you can assign logical names to disk and tape devices when you mount them, and then use the logical name rather than the device name in subsequent commands. If you do not specify a logical name, the MOUNT command assigns the default logical name DISK$TEST_FILES to the device DMA2.

3.3.1 Requesting Operator Assistance

At some installations, operators are provided to perform the physical chores of mounting and dismounting both system and private disk and tape volumes. When this is the case, you can:

- Allocate a device of the required type.
- Send a message to the operator specifying the name of the device you allocated and the volume you want mounted. Give the physical identification on the outside of the volume and tell the operator where the volume is.
- Wait until the operator responds with a message indicating that the volume is mounted.

The REQUEST command sends a message to an operator. If your installation has more than one operator, they may be designated for specific functions -- one operator to handle requests for disks, another for tapes, and so on. Options for the REQUEST command qualifier /TO -- for example, DISKS and TAPES -- route your request to the proper operator.

Assume, for this example, that there is an operator designated to mount and dismount disks. To request this operator to mount the volume TEST_FILES on the device DMA2, you could issue the command:

```
$ REQUEST/TO=DISKS/REPLY -
$_"Please mount TEST_FILES (shelf slot 6B) on DMA2"
```

The /REPLY qualifier indicates that you want to wait until the operator completes the request.

You receive the message:

```
%OPCOM-S-OPRNOTIF, operator notified, waiting...13:14:26
```

The operator locates the physical volume, places it on the device, and types a message indicating that the request is satisfied. Then, you receive the messages:

```
%OPCOM-S-RQSTCMPLTE, request complete
%OPCOM-S-OPREPLY, "Go..."
```

The text of the second message optional text typed by the operator. Now, you can mount and begin using the volume.

3.3.2 Dismounting Volumes

When you no longer need access to the files on a volume, you can release the volume with the DISMOUNT command. For example:

```
$ DISMOUNT DMA2:
```

The DISMOUNT command ensures that all files on the volume are closed before the dismounting is complete.

By default, the DISMOUNT command also unloads the volume on the device and makes the device not ready.

If you allocated the device before using it, and no longer need the device, deallocate it so that other users can obtain access to it. The DEALLOCATE command deallocates the device. For example:

```
$ DEALLOCATE DMA2:
```

If you had operator assistance in mounting the volume, remember to request the operator to physically dismount the volume and return it to its place.

3.4 USING DISK AND TAPE VOLUMES

When a volume has been mounted, you can execute programs that perform input/output operations to the volume, or you can use DCL commands to read and write files. Note the following restrictions on the use of DCL commands for files on disk and tape volumes:

- The PRINT and SUBMIT commands cannot access files on allocated devices. You can print or submit files on private disk volumes if you mount the volume as a shareable volume. To print or submit a file on a tape volume, you must copy the file to a shared disk volume.

- The following commands cannot process files on magnetic tapes:

DELETE	PURGE
DIFFERENCES	RENAME
LIBRARY	RUN
LINK	SET PROTECTION
	UNLOCK

- You can execute a command procedure that exists on a magnetic tape volume, as long as the procedure does not invoke other procedures and does not issue any GOTO commands referring to labels that precede the command.

Note, also, that you cannot use DCL commands to read and write files that are not in the standard formats supported by VAX/VMS (these formats are described in greater detail, below). To execute programs to read and write files not in the standard format, you must mount the volume with the /FOREIGN qualifier. The following sections provide complete examples of the steps to prepare and use disk and tape volumes for file storage and to access existing files. The examples show how to prepare and use RK06/RK07 disk packs and magnetic tapes; however, the procedures outlined are applicable to other devices as well.

3.4.1 Using Disks

Disks are random-access devices, and files must be cataloged in directories. Therefore, after you initialize a disk, you must create a directory before you can write any files on the disk volume.

The following example shows how to allocate an RK06/RK07 device, initialize and mount a volume on it, create a directory, and write files on the volume.

```
$ SHOW DEVICES DM:
Device    Device      Device
Name      Status Characteristics
DMA0:   on line MNT
DMA1:   on line MNT ALL
DMA3:   on line

$ ALLOCATE _DMA3:
  _DMA3:  ALLOCATED
```

This example illustrates the SHOW DEVICES command; the command requests a display of the current status of RK06/RK07 devices. The response from the command indicates that the device named DMA3 is available. The ALLOCATE command allocates the device so you can place the volume on it.

```
$ INITIALIZE DMA3:  PUBS_BACKUP -
$_/PROTECTION=(SYSTEM:RWED,OWNER:RWED,GROUP:RWED,WORLD:R)
$ MOUNT DMA3:  PUBS_BACKUP
$ CREATE/DIRECTORY DMA3:[PUBS]
$ ASSIGN DMA3:[PUBS]  P:
$ COPY *.*  P:
$ COPY [PRIMER]*.*  P:
$ COPY [COMMANDS]*.*  P:
```

The INITIALIZE command initializes the volume and writes the label PUBS_BACKUP on it. The protection code allows group members and users with system UICs all access and restricts access by all other users to reading. The MOUNT command mounts the volume. The CREATE command with the /DIRECTORY qualifier creates a directory file named PUBS on the device DMA3. The COPY commands copy the highest versions of all files in the current default directory and in the directories PRIMER and COMMANDS to the directory just created.

Note the use of the ASSIGN command to assign a logical name, P, to the device and directory name on the RK06/RK07 volume.

3.4.1.1 **Copying Files from Files-11 Structure Level 1 Disks** - The default format for files on disks is called Files-11 Structure Level 2. You can also initialize disks in the Files-11 Structure Level 1 format. Structure Level 1 is the format used by other DIGITAL operating systems, including RSX-11M, RSX-11D, and IAS.

To initialize a Structure Level 1 disk, use the /STRUCTURE=1 qualifier on the INITIALIZE command. When you use the MOUNT command to mount the volume, the MOUNT command will internally identify the volume as a Structure Level 1 volume. Subsequently, all commands you use to access files (COPY, DELETE, and so on) will default the file format to Structure Level 1 automatically. Note that directories on Structure Level 1 disks must have names in UIC format to be readable by RSX-11 or IAS.

3.4.1.2 **Sharing Volumes** - System disk volumes containing users' default directories and other volumes containing files belonging to more than one user are designated, at the time that they are mounted, as shareable. The devices on which these volumes are mounted are not allocated, thus allowing access by other system users. These volumes are physically loaded and mounted by system operators and managers with the /SYSTEM qualifier of the MOUNT command; you need not issue an explicit MOUNT command to access files on these volumes (you must, however, have access privileges as defined in the volume's protection code).

Other volumes can be designated as shareable among many users; use the /SHARE qualifier on the MOUNT command when you want to mount a device for other users to access. For example:

```
$ MOUNT/SHARE DMA3:  PUBS_BACK
```

This MOUNT command indicates that other users can access the volume PUBS_BACK. Each user who wants to access the volume must also issue a MOUNT command with the /SHARE qualifier. Note that if the device was allocated before the MOUNT command was issued, the /SHARE qualifier deallocates the device.

Access to the volume is restricted according to the protection code on the volume. The system identifies the volume by its volume label, rather than by the device on which it is mounted. Other users who want to access the volume do not need to know the physical name of the device, but only the generic device type and volume label.

For example, another user who wants to use the volume mounted in the example above would issue the MOUNT command, as follows:

```
$ MOUNT/SHARE  DM:  PUBS_BACK PUBS
```

Thereafter, the device can be referred to by the logical name PUBS; the sharer does not need to know the name of the physical device.

Because the system uses volume labels to identify shared volumes, two volumes that have the same label cannot be mounted with the /SHARE (or /SYSTEM or /GROUP) qualifier at the same time.

3.4.2 Using Tapes

The default format for reading and writing tapes is the ANSI X327-1977 Magnetic Tape Labels and File Structure for Information Interchange standard.

The following examples show how to allocate, initialize, and use a tape to backup your disk files. The procedures are similar to those outlined above for using disk volumes. However, tapes are sequential access devices and do not have directories.

```
$ ALLOCATE MT:
  _MTA2:  ALLOCATED
```

The ALLOCATE command requests the allocation of any tape device; the response indicates that the device MTA2 was available and is now allocated to you. You can now physically load the tape on the drive. Next, initialize the tape:

```
$ INITIALIZE  MTA2:  GMB001 -
$_/PROTECTION=(GROUP:R,WORLD)
```

The INITIALIZE command specifies the device name (MTA2) and the volume label for the tape volume (GMB001). The /PROTECTION qualifier defines a protection code restricting group access to read and allowing no access to the world. You can now use the MOUNT command to mount the volume and write files to it:

```
$ MOUNT  MTA2:  GMB001
%MOUNT-I-MOUNTED, GMB001 mounted on _MTA2:
$ COPY *.*  MTA2:
```

The MOUNT command specifies the device name and volume label of the volume on the device. The COPY command copies the highest versions of all files in your default directory onto the tape. The file names and file types of the output files default to the same file names and file types as the input files.

To verify that the files were successfully copied, you can use the DIRECTORY command:

```
$ DIRECTORY MTA2:
```

The DIRECTORY command lists the file names and file types of all files on the tape.

When you have finished using the tape, dismount it and deallocate it, as shown below:

```
$ DISMOUNT MTA2:
$ DEALLOCATE MTA2:
```

If you do not dismount and deallocate the tape, the system does so automatically when you log out.

3.4.2.1 **Reading and Writing Tape Files** - A magnetic tape is a sequential device. With DCL commands, you can write new files only at the end of existing data on the tape and you cannot delete files from a tape. You can, however, append data at the end of an existing file on a tape; all files that follow the appended file are overwritten. The APPEND command will not, however, overwrite a file that has not expired. If you want to overwrite a tape completely, you must re-initialize it.

When you want to copy files from an existing tape, you can selectively copy files from the tape by specifying the file name and file type of the file you want to copy, as shown below.

```
$ MOUNT MTB2:  GMB001
$ COPY MTB2:ASTRO.SRC  ASTRO.OLD
   .
   .
$ COPY MTB2:ASTRO.FOR  ASTRO2.FOR
$ DISMOUNT MTB2:
```

The COPY commands copy specific files from the tape. After each copy request, the COPY command leaves the tape positioned at the end of the file it has just copied. When it looks for the next file, it continues searching until the end of the tape; if it does not find the file, it rewinds the tape and searches until it either locates the file or returns to the point on the tape at which it started.

3.4.2.2 **Version Numbers for Tape Files** - Files that you write onto tape with DCL commands are written in sequential order and all have version numbers of 0. If you want to read a file from a tape and you do not specify a file version number, the command locates the next file with that file name and file type that is physically on the tape.

If you write a file onto a tape and request the next highest version number, the version number associated with the file is 0.

3.4.3 Multivolume Tape Sets

The VAX/VMS operating system allows you to create and access files that span more than one physical tape volume. Each volume in a multivolume set has a unique volume label and a relative volume number within the set.

Processing of multi-volume files requires the attendance of an operator or user to respond to requests to switch volumes; when a command or program attempts to read or write beyond the end of a tape,

the system automatically sends a message to the system operator's console (or to a terminal designated as an operator's terminal). The message indicates:

- The device name
- The relative volume number of the next volume in the volume set
- The label, if known

Before processing can continue, the operator must physically mount the volume, place the device online, and type a reply to the message. If no operator is available, you can load the volume yourself and reply to the message from the operator's console or an operator's terminal.

When you issue the MOUNT command to begin processing a multivolume file, you can specify the labels on each of the volumes in the MOUNT command. For example, after physically loading the first volume on the device MTA0, you can issue the MOUNT command as follows:

```
$ MOUNT MT:  GMB001, GMB002, GMB003
```

The MOUNT command verifies the label on the volume and returns a message indicating successful completion if the volume label is correct.

Subsequently, when the tape reaches end-of-tape, the system automatically sends a request to the system operator to mount the volume labeled GMB002. The messages the operator receives typically look like the following:

```
Opcom, 02:18:48.01, GEOFF      Accnt=TEXTPROC  Reply-ID=131084
Opcom, MOUNT relative volume 2 (GMB002) on MTA0:
```

After loading the volume and readying the device, the operator types the message:

```
$ REPLY/TO=131084
```

The system verifies the reply-ID and then verifies the volume. If the correct volume is mounted, processing continues.

At the end of that tape, it sends a message to mount the label GMB003. No more explicit MOUNT commands are required.

3.4.3.1 **Creating a Multivolume Tape Set** - When you initially create a file that spans tape volumes, you may not know that multiple volumes are required. If, while you are writing to the tape, the tape reaches end-of-tape, the system suspends processing to notify the operator that a new volume is required. In this case, the system does not know the volume label. The operator must mount a volume and enter the volume label on the REPLY command as shown below.

```
$ REPLY/TO=196620 "GMB004
```

If the tape is a new tape, operator can request that it be initialized by specifying /INITIALIZE following the label. For example:

```
$ REPLY/TO=196620   "GMB004/INITIALIZE
```

The system performs normal protection and expiration checks before initializing the volume. The operator can also specify /BLANK to override the checking for protection information on tapes that have been read by a verifying machine.

3.4.3.2 **Using Multiple Tape Drives** - You can overlap the mounting of volumes in a multivolume set by specifying more than one drive in the MOUNT command. For example:

```
$ MOUNT  MTA0:,MTA1:  GMB001, GMB002, GMB003
```

The MOUNT command verifies the volume on MTA0. If the volume GMB002 is located on the device MTA1 when the end-of-volume is reached on GMB001, processing continues. However, when the end-of-volume occurs on GMB002, the first volume is dismounted from MTA0 and the system sends a message to the operator to MOUNT the third volume on MTA0.

3.5 ACCESSING DEVICES IN BATCH JOBS

You can write command procedures to mount volumes, execute DCL commands or your own programs to write or access files on the volume. By using logical names to refer to devices and files, you can use the same command procedures without modification each time you want to access a volume.

For example, if you use the same RK06 disk pack to back up your files on a weekly basis, you can submit as a batch job a command procedure like the following:

```
$ TRY:
$ ALLOCATE  DM:  RK
$ IF  $STATUS  THEN  GOTO  OKAY
$ WAIT  00:05
$ GOTO  TRY
$ OKAY
$ REQUEST/REPLY /TO=DISKS  -
"PLEASE MOUNT  RK06 BACK_UP_GMB ON ''F$LOGICAL("RK")' "
$ MOUNT  RK  BACK_UP_GMB
$ COPY/REPLACE  *.*  RK:*.*
$ DIRECTORY/FULL/OUTPUT = BACKUP.LOG  RK:
$ DISMOUNT RK
$ PRINT  BACKUP.LOG
$ DEALLOCATE  RK
$ REQUEST/TO = DISKS    "All done, thanks..."
```

In this job, the procedure places itself in a wait state if no RK06/RK07 is available and loops to repeat the request. When the ALLOCATE command completes successfully, a message is sent to the operator.

The logical name, RK, assigned on the ALLOCATE command is used in all subsequent commands. The lexical function F$LOGICAL is used in the REQUEST command; this function translates the logical name RK and substitutes the equivalence name in the message displayed at the operator's console.

When the REQUEST command notifies the operator to mount the correct volume, the job waits until the operator responds before continuing.

For more information on how to create and execute command procedures, see Chapter 5, "Command Procedures and Batch Jobs."

CHAPTER 4

PROGRAMMING WITH VAX/VMS

The VAX/VMS operating system provides concurrent time-shared multiprogramming and batch job processing: many users can be logged in at terminals to create and test new programs and applications interactively at the same time that production applications and time-critical process control applications are running.

This chapter describes:

- Commands for program development
- Debugging programs
- Exit handlers and condition handlers
- Process concepts

Table 1-5 in Section 1.10, "Summary of VAX/VMS DCL Commands" lists the commands described in this chapter. For details on the parameters and qualifiers for any of these commands, see the command descriptions in Part II.

4.1 COMMANDS FOR PROGRAM DEVELOPMENT

Figure 4-1 illustrates the steps required to create and execute programs in VAX/VMS.

The following example illustrates the DCL commands you would use to create, compile, link, and execute a FORTRAN program named AVERAGE using DCL commands:

```
$ EDIT AVERAGE.FOR
Input:DBA1:[MALCOLM]AVERAGE.FOR;1
00100
    .
    .
    .
    input source statements
    .
ESC
*e
$ FORTRAN AVERAGE
$ LINK AVERAGE
$ RUN AVERAGE
```

Use the *editor* to create a disk file containing your source program statements. Specify the name of this file when you invoke the compiler or assembler.

The *FORTRAN* and *MACRO* commands invoke language processors that check syntax, create object modules, and if requested, generate program listings.

If a processor signals any errors, use the editor to correct the source program.

The *linker* searches the system libraries to resolve references in the object module and create an executable image. Optionally, you can specify private libraries to search, and request the linker to create a storage map of your program.

The linker issues diagnostic messages if an object module refers to subroutines or symbols that are not available or undefined. If the linker cannot locate a subroutine, you must reissue the LINK command specifying the modules or libraries to include. If a symbol is undefined, you may need to correct the source program.

The *RUN* command executes a program image. While your program is running, the system may detect errors and issue messages. To determine if your program is error-free, check its output.

If there is a bug in your program, determine the cause of error and correct the source program.

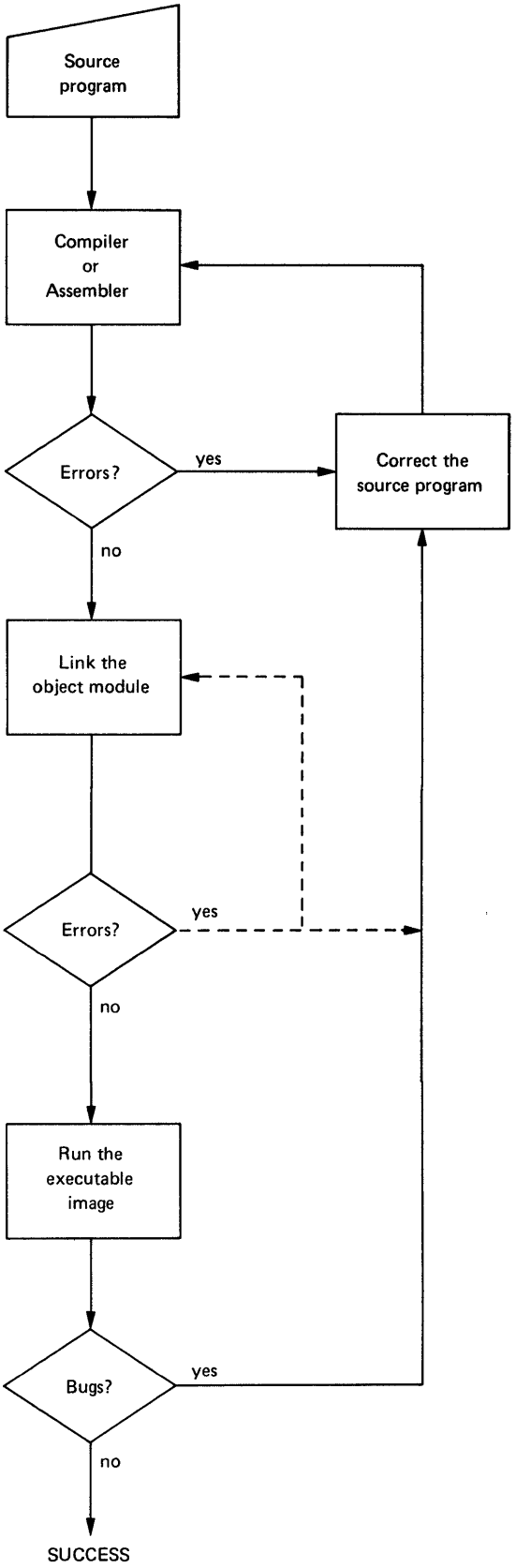


Figure 4-1 Steps in Program Development

Note that the EDIT command invokes the default system editor, SOS. SOS prompts for input lines until you use (ESC) to terminate input. Then, the E (Exit) command requests SOS to write the input data onto disk.

The FORTRAN command invokes the VAX-11 FORTRAN IV-PLUS compiler to compile the source statements.

If you are a VAX-11 MACRO programmer, you would give the input file a file type of MAR, and use the MACRO command to assemble the input file.

All of the commands, EDIT, FORTRAN, LINK, and RUN, are shown in their simplest forms, without qualifiers. By giving the source file a file type of FOR in the file creation, you can allow the file types of the input and output files to assume the defaults for the FORTRAN, LINK, and RUN commands.

You can find a description of each of these commands, and lists of the valid qualifiers, in Part II of this manual.

The next few sections discuss DCL commands and system programs that can help you in developing, testing, and maintaining your programs.

4.1.1 Program Libraries

The LIBRARY command creates and maintains libraries of object modules or macros. A library is a file that contains its own directory listing the entries it contains.

4.1.1.1 Object Module Libraries - Object module libraries are convenient for storing routines that are called frequently by many programs. For example, if you have compiled the object modules named TIMER, CALC, and SWITCH, you could create a library named COMMON.OLB using the LIBRARY command as follows:

```
$ LIBRARY/CREATE  COMMON  TIMER,CALC,SWITCH
```

When you issue a LINK command to link an object module that calls any of these routines, you can specify the library as a linker input file using the /LIBRARY qualifier:

```
$ LINK  TESTPROG,COMMON/LIBRARY
```

When the linker links the module TESTPROG, it automatically searches the library COMMON.OLB when it encounters any undefined external references.

The system library STARLET.OLB contains system routines that are called frequently. The linker automatically searches this library after searching any private libraries you specify for undefined external references.

4.1.1.2 **Macro Libraries** - If you are a MACRO programmer, you can also use the LIBRARY command to catalog macros that you use frequently. For example, to create a macro library from the macros contained in the files DESCRIPTOR.MAR, TRANSLATE.MAR, and RANDOM.MAR, issue the command:

```
$ LIBRARY/CREATE/MACRO  LOCALMAC  DESCRIPTOR,TRANSLATE,RANDOM
```

To assemble a program that invokes any of these macros, specify the name of the library on the MACRO command with the /LIBRARY qualifier, as shown below:

```
$ MACRO LOCALMAC/LIBRARY+RUNTEST /OBJECT
```

This MACRO command indicates that there are two input files: LOCALMAC.MLB, a library; and the source file, RUNTEST.MAR.

The system library STARLET.MLB contains system macros. The assembler automatically searches this library to locate macro definitions after searching any private libraries you specify, as in the above example.

4.1.2 Controlling Program Updates and Modifications

VAX/VMS provides several commands and programs you can use to track and control updates that you make to your programs.

4.1.2.1 **Updating Source Programs** - To update or modify a source program, you can use the interactive editor, SOS. Because SOS, by default, creates a new file (with a higher version number) each time you edit a file, you can keep previous versions of a file for back up. SOS does not, however, provide you with a record of the changes that you have made.

VAX/VMS also provides a batch editor, SLP. With SLP, you can insert, delete, or replace lines in a file and create a new file incorporating your changes. SLP also creates a record of all the modifications that you made.

Both SOS and SLP are invoked with the EDIT command. Both of these editors are described in detail in the VAX-11 Text Editing Reference Manual.

4.1.2.2 **Comparing Versions of Files** - The DIFFERENCES command invokes a file comparison program that compares the contents of two files to determine what differences, if any, exist between them. DIFFERENCES creates an output file that summarizes the differences.

The DIFFERENCES command is described in Part II.

4.2 DEBUGGING

The VAX-11 Symbolic Debugger is an interactive debugging program. It has an extensive set of commands that allow you to examine and modify a program in memory while it is executing.

The debugger uses three sets of information:

- Local symbol table information
- Global symbol information
- Traceback information

When you use DCL commands to compile or assemble and link a program, you can control what information, if any, you want to include in the image.

If you request local symbol table information you can refer to actual variable names and statement labels when you issue DEBUG commands. If you request traceback information, the debugger can trace the call stack when an error occurs during image execution.

By default, the FORTRAN IV-PLUS compiler and the VAX-11 MACRO assembler generate traceback information, but not local or global symbol information, in the object module. The linker, also by default, includes the traceback information in the image file so that you receive a symbolic traceback when an error occurs.

4.2.1 Symbolic Debugging

If you want to use the complete symbolic capabilities of the debugger, you must request the debugger when you compile and when you link a program. The following example shows the commands to compile and link a FORTRAN program with the debugger:

```
$ FORTRAN/DEBUG/NOOPTIMIZE  PRECIP
$ LINK/DEBUG  PRECIP
$ RUN PRECIP

     DEBUG Version 1.0
%DEBUG-I-INITIAL, language is FORTRAN, scope and  module  set  to
'PRECIP'
DBG>
```

The FORTRAN command requests that the debugger symbol table and traceback information be included in the object module (the /NOOPTIMIZE qualifier ensures a one-to-one correspondence between the source program statements and the machine code in the object module).

The LINK command requests inclusion of the debugger and local and global symbol definitions in the image.

When you issue the RUN command to execute an image linked with the debugger, the debugger receives control, identifies itself, and prompts for you to begin entering DEBUG commands. If you do not want the debugger to prompt when you execute an image, issue the RUN command as follows:

```
$ RUN/NODEBUG PRECIP
```

In this case, the debugger does not prompt; however, you can invoke it by issuing the DEBUG command when an error occurs.

For complete descriptions of the DEBUG commands and considerations for using the debugger, see the VAX-11 Symbolic Debugger Reference Manual.

Note that symbol table information and, to a lesser extent, traceback information increase the size of an object module. When you have debugged a program, you can recompile or reassemble without the symbol table information, retaining traceback information in the event of unexpected errors in the future.

You can also exclude traceback information from modules that you catalog in object module libraries. Otherwise, the traceback information is included in all modules that link with the library module.

4.2.2 Debugging with Virtual Addresses

You can debug an image that was not compiled and linked with the debugger symbol table. The /DEBUG qualifier on the RUN command requests the debugger at run time. For example:

```
$ RUN/DEBUG ORION
```

To specify memory locations for the debugger, you must have both a machine code listing of the object module and a full map from the linker. The map gives the virtual address (in hexadecimal) of each module, global symbol, and PSECT in the image.

If you do not link or run an image with the debugger, you can debug a program using virtual addresses with the DCL commands EXAMINE and DEPOSIT. The EXAMINE command displays the current contents of a location or range of locations; the DEPOSIT command modifies a location. These commands provide a useful, but limited, debugging capability when you need to debug a program that has not been linked or run with the debugger. The DEPOSIT and EXAMINE commands are described in detail in Part II.

4.2.3 Interrupting Program Execution

When you use the RUN command to execute an image interactively, you cannot execute any other images or DCL commands until the image completes; that is, until the image exits. If you enter a command line, the system saves the line in the terminal type-ahead buffer, but does not echo the line or begin to execute the command until the image exits.

If you need to interrupt an image while it is executing, press CTRL/C or CTRL/Y. Generally, the effect of both of these CTRL key sequences is the same: the image is interrupted (but unchanged), the type-ahead buffer is purged, and the command interpreter receives control.

Note: Some system or application programs may contain special routines coded to receive control when CTRL/C is entered. If you use CTRL/C to interrupt such a program, the CTRL/C handling routine receives control, rather than the command interpreter. Use CTRL/Y to bypass a CTRL/C handling routine when you want to interrupt a command or program to enter a DCL command. If you use CTRL/C to interrupt a command or program that does not have a CTRL/C handling routine, then CTRL/C has the same effect as CTRL/Y and echoes as ^Y. For information on how to code a CTRL/C handling routine, see the VAX/VMS I/O User's Guide.

The following example shows CTRL/C or CTRL/Y being pressed to interrupt a program that is looping:

```
$ RUN NAMETST
ENTER YOUR NAME:
ENTER YOUR NAME:
ENTER YOUR NAME:
^Y
$
```

The dollar sign ($) prompt indicates that you can enter a DCL command. After you have interrupted an image (or a DCL command) with CTRL/Y, you can:

- Terminate the image, that is, force it to exit by entering the STOP command or by entering a RUN command to execute another image.
- Issue the CONTINUE command to resume execution of the image. The image resumes execution from the point of interruption.
- Issue the DEBUG command, if the image was linked or run with the /DEBUG qualifier. The DEBUG command gives control to the debugger.

You can also issue any other DCL command. Most DCL commands you enter at CTRL/Y level have the same effect as the RUN command; that is, the current image is forced to exit before the command can be executed. However, the following commands are performed within the command interpreter and thus do not affect the image that was interrupted:

```
=
ALLOCATE        GOTO                        SHOW TIME
ASSIGN          IF                          SHOW DEFAULT
CLOSE           INQUIRE                     SHOW PROTECTION
DEALLOCATE      OPEN                        SHOW STATUS
DEASSIGN        READ                        SHOW SYMBOL
DEFINE          SET DEFAULT                 SHOW TIME
DEPOSIT         SET PROTECTION/DEFAULT      SHOW TRANSLATION
EXAMINE         SET VERIFY                  WAIT
EXIT            SET UIC[1]                  WRITE
```

For example, you could interrupt any command or program, issue the SHOW TIME command, and then continue the image, as follows:

```
$ RUN GRADES
^Y
$ SHOW TIME
  07-JUN-1978 10:54
$ CONTINUE
```

Note that you can also use CTRL/O, CTRL/S, and CTRL/Q to interrupt, suppress, or continue terminal output from a program image, just as you can for the output of a DCL command. These control key functions do not affect the program image.

---

1 This command is described in the VAX/VMS Operator's Guide.

4.3 EXIT HANDLERS AND CONDITION HANDLERS

Programs written in VAX-11 FORTRAN IV-PLUS and VAX-11 MACRO can take advantage of operating system services that allow an image to respond to special situations.

4.3.1 Exit Handlers

When you write a program to execute on VAX/VMS, you can provide the program with an exit handler. An exit handler is a special routine that receives control when the image exits. The exit handler can determine whether the image is exiting normally; that is, whether the program completed successfully, or exited as the result of an error condition.

If you have used the RUN command to execute an image that has an exit handler, and you interrupt the image with CTRL/Y, you can control whether the exit handler is actually given control. If you issue the STOP command, the exit handler is not executed. If, on the other hand, you issue another DCL command or the RUN command to execute another image, the exit handler in the interrupted image is allowed to execute before the specified command is performed.

For information on how to code an exit handling routine, see the VAX/VMS System Services Reference Manual.

4.3.2 Exception Conditions

The system interrupts image execution when the image incurs an exception condition; that is, it attempts an illegal operation. Some examples of situations that cause these conditions, called exceptions, are:

- Arithmetic overflow or underflow
- A memory access violation
- An invalid operation code
- An invalid argument list to the math library

When an exception occurs, the system searches for a routine that can respond to the condition; these routines, called condition handlers, can be declared from user programs. If no user-declared condition handlers are located, or if a user-declared condition handler cannot respond to a particular situation, the system gives control to a default handler. This handler terminates the image; obtains as much information as it can about the exception condition, including any available traceback information; and summarizes the information for you.

4.4 PROCESS CONCEPTS

The executable program image created by the linker executes within the context of the process created for you at login. This process can execute, serially, many different images during your terminal session. When you issue the LOGOUT command to end your terminal session, the system deletes the process.

The VAX/VMS operating system manages all users' requests to execute images in terms of the processes issuing the requests. The system provides each process with a unique process identification number (PID), a character string process name, and a distinct environment. Figure 4-2 illustrates the relationship of a terminal user to the process that the system creates and the images executed in the process by the RUN command.

4.4.1 Priorities, Privileges, and Quotas

The characteristics of an individual process -- that is, the process's context -- determine the nature of the images that the process will be allowed to execute. Most of these characteristics are obtained from the user authorization file. When you log in, the defaults established for you by the system manager are associated with the process that the system creates for your terminal session.

These defaults determine:

- The base execution priority given to the images that the process executes. In general, communications and process control applications are given higher priorities than batch jobs and interactive users.
- Resource quotas that limit or restrict the frequency or amount of a particular system resource any image can use. For example, an open file quota limits the number of files a process can have open at any one time.
- User privileges to perform system functions or to call specific system services. For example, the ability to place names in the group or system logical name tables is controlled by a privilege.

The base priority, resource quotas, and privileges granted to general users are adequate for time-sharing program development requirements. In fact, many of the DCL commands you execute perform privileged functions on your behalf, so you do not require the privilege. Note, however, that for some commands, you must have a user privilege to use a particular qualifier. These restrictions are noted in the command descriptions, as appropriate.

You can determine the current privileges and quotas available to your process by issuing the following command:

```
$ SHOW PROCESS/PRIVILEGES/QUOTAS
```

Tables 1-7 and 1-8 list the privileges and quotas defined by the VAX/VMS operating system.

4.4.2 Input, Output, and Error Streams

The base priority, quotas, and privileges assigned to a process at its creation are only one aspect of the context created at login. The system also establishes equivalences for default process logical names, including SYS$INPUT, SYS$OUTPUT, and SYS$ERROR. These logical names provide permanent associations for the process's input, output and error streams. For an interactive process, these logical names are initially equated to the terminal, and are used by the command interpreter to read command lines and to display output and error messages.

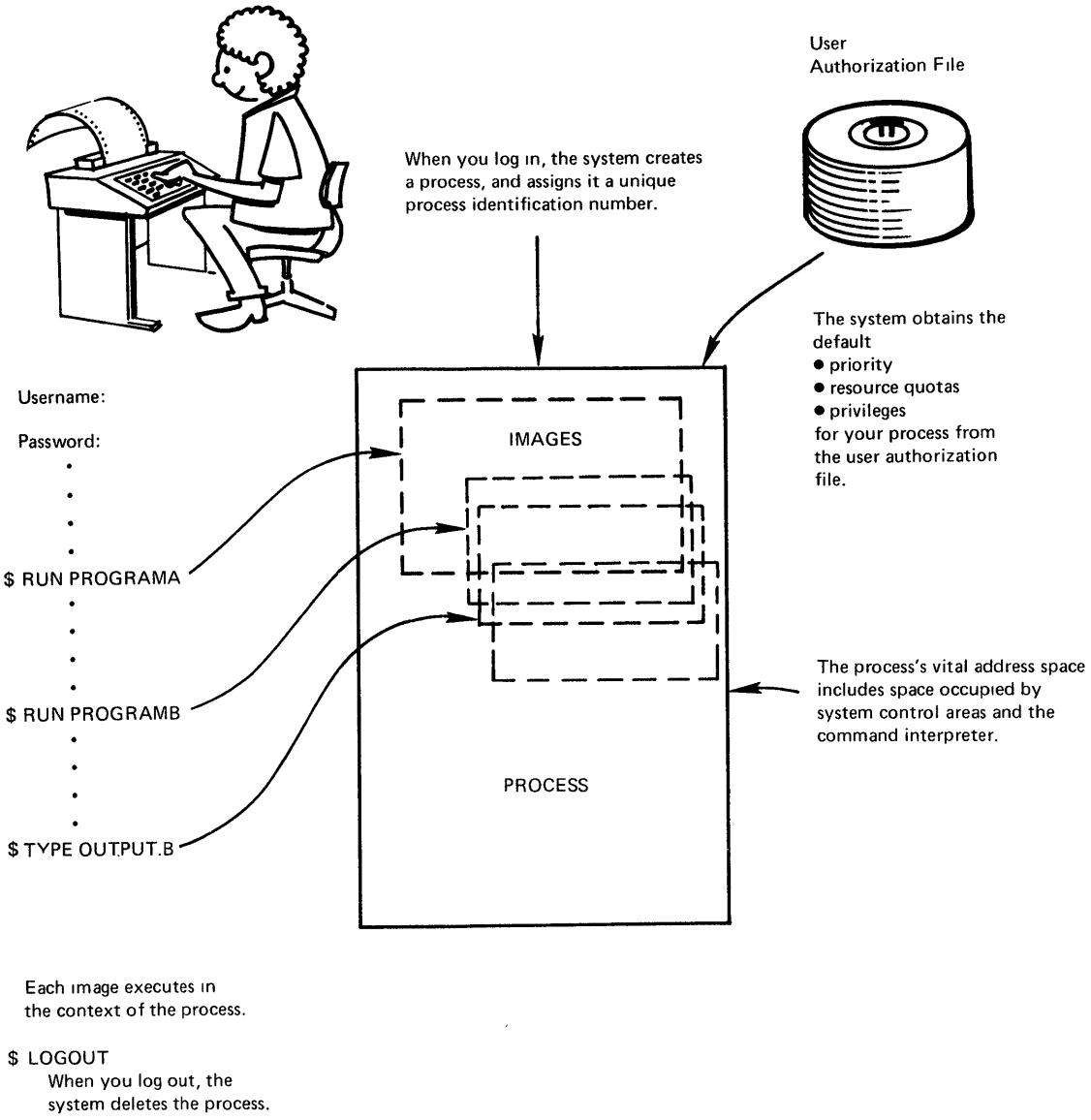


Figure 4-2 An Interactive Process

These logical names are also available to all images that the process executes. For example, a program that performs explicit input/output requests through RMS (Record Management Services) macros or system services can write records to SYS$OUTPUT. If you execute this image interactively, the output is directed to your current terminal; if you execute the image in a batch job, the output is directed to the batch job output log.

4.4.3 Processes and Subprocesses

A process can execute only one program image at a time. Some applications may require concurrent execution of cooperating programs. Because the system uniquely identifies every process, an image executing in one process can communicate with or control another process by referring to that process's identification number.

Processes executing with the same group number in their UICs can also refer to one another by process name, that is, a character string name assigned to the process. When you first log into the system, the system gives your process the same name as your user name. If you log in at more than one terminal, processes after the first are given names based on the name of the terminal at which you logged in.

Most processes in the system are detached processes; that is, they execute independently of one another. The process that the system creates for you at login is a detached process. An image executing in a process can create another type of process, called a subprocess. The process that creates a subprocess is called the owner process.

The owner of a subprocess can:

- Define the base priority, privileges, and resource quotas that the subprocess will have. Some of these resource quotas are deducted from the owner's quotas.
- Specify the name of an executable image to be executed in the subprocess, and control the execution of the image.
- Obtain information about the status of the subprocess and the system resources it has used.

Generally, a subprocess executes a single image, and when the image exits, the system deletes the subprocess. If the owner process is deleted (for example, if you log out) and a subprocess still exists, the system also deletes the subprocess.

If you plan to develop an application to use cooperating processes, you should be familiar with the VAX/VMS system services that provide process communication and control functions. These services are described in detail in the VAX/VMS System Services Reference Manual.

In conjunction with the system services that control processes and subprocesses, you can use the RUN command to create a subprocess to execute a particular image. For details, see the description of the RUN (Process) command in Part II.

CHAPTER 5

COMMAND PROCEDURES AND BATCH JOBS

A command procedure is a file containing DCL commands, command or program input data, or both. You can use command procedures to catalog sequences of commands you frequently use during an interactive terminal session; or, if you are a batch user, to submit all your jobs for processing.

In its simplest form, a command procedure consists of two or more command lines that the command interpreter executes. In its most complex form, a command procedure resembles a program written in a high-level programming language: it can establish loops and error checking procedures; call other procedures; pass values to other procedures and test values set in other procedures; perform arithmetic calculations and input/output operations; and manipulate character string data.

This chapter describes:

- Creating command procedures
- Executing command procedures interactively
- Submitting batch jobs
- Command levels
- The LOGIN.COM file
- Using symbols in command procedures
- Commands to control the execution of a command procedure
- Reading and writing files
- Lexical functions

Table 1-6 in Section 1.10 "Summary of VAX/VMS DCL Commands" lists the commands described in this chapter. For complete details on the parameters of any of these commands, see the command descriptions in Part II.

5.1 CREATING COMMAND PROCEDURES

You can create a command procedure using any method or media available to you: if you are an interactive terminal user, use the CREATE command or the editor to create the command procedure interactively; if you are a batch job user, punch the cards containing the command procedure using a card punch.

Each line in a command procedure represents a line that you want the system to process. You enter the lines into the file in the order in which you want the system to process them. For example, to create a command procedure named TESTALL.COM that contains RUN commands to execute the program images named TESTA.EXE, TESTB.EXE, and TESTC.EXE, you would create a file containing the following lines:

```
$ RUN TESTA
$ RUN TESTB
$ RUN TESTC
```

Note that you must precede each line containing a DCL command (including comment lines) with a dollar sign ($); you can precede or follow the dollar signs with none or one or more blank spaces or tabs.

If you continue any command on more than one line using the continuation character, a hyphen (-), the lines after the first line must not contain a dollar sign. For example:

```
$ PRINT TEST.OUT -
   /AFTER=18:00 -
   /COPIES=10 -
   /QUEUE=LPB0:
```

The qualifiers for this PRINT command are placed on separate lines in the command procedure for readability, with continuation characters to indicate that the command is entered on more than one line. The spaces preceding each qualifier are not required, but are also included to make the command procedure more readable.

5.1.1 Entering Data in Command Procedures

When the system executes a command procedure, either interactively or as a batch job, it associates the command procedure file with the logical name SYS$INPUT. Therefore, if a command procedure executes a DCL command or a program that reads from SYS$INPUT, the input is actually read from the command procedure.

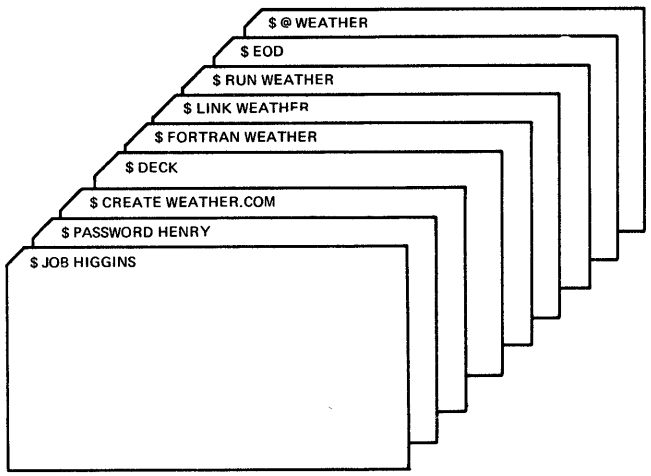
For example, when you issue the CREATE command, the system reads input lines for a file from the command input stream. If you issue the CREATE command interactively, the command input stream (that is, SYS$INPUT) is your terminal. You indicate the end of the data stream by pressing CTRL/Z.

When you use the CREATE command in a command procedure file, the input stream is the command procedure; thus, you place the input for the file specified in the command procedure, as follows:

```
$ CREATE WEATHER.DAT
JAN  39   3
FEB  42   1
MAR  50  20
   .
   .
   .
DEC  46  25
$ RUN WEATHER
```

In a command procedure, the end of the input data for the CREATE command (or any command or program that is reading input data) is indicated by a line that begins with the dollar sign character, or by the physical end-of-file of the command procedure. In the above example, the end of input is signaled by the RUN command line.

If any data lines you place in the input stream begin with dollar signs ($), you must delimit the data stream with the DECK and EOD (End of Deck) commands. This technique is particularly useful for batch users who submit all work through the system card reader. For example, if you use the CREATE command to write a command procedure into a disk file, you would use the DECK and EOD commands as shown below:



This procedure creates and executes the procedure WEATHER.COM.

5.1.2 Command Procedures without Commands

You can construct command procedures that contain only data to be read by a command or program; or that contain only qualifiers or parameters for a command; or both. When you specify the @ in any place in a command, the command interpreter begins reading input from the procedure file. For example, you could create a command procedure that contains a number of qualifiers you frequently use together when you issue a LINK command, as shown below:

```
/DEBUG -
/SYMBOL_TABLE -
/NOSUPPRESS=SYSLIB -
/MAP/FULL/CROSS_REFERENCE
```

If this procedure is named DEFLINK.COM, you can subsequently request these qualifiers on a LINK command line by executing the procedure as follows:

```
$ LINK  SYNAPSE@DEFLINK
```

Note that no space precedes the at sign (@) in the above example. If you type a space before the at sign, the command interpreter assumes that the command file contains a parameter for the LINK command. Because the LINK command allows only one parameter, an error would result.

5.2 EXECUTING COMMAND PROCEDURES INTERACTIVELY

When you execute a command procedure, give the file specification of the command procedure following the at sign. The @ command assumes current disk and directory defaults, and the default file type of COM. For example, to execute the command procedure WEATHER.COM in your default disk directory, issue the command:

```
$ @WEATHER
```

To execute a command procedure that is not in your default disk directory, give the complete file specification as follows:

```
$ @DBB2:[COMMON]SETUP.FIL
```

This @ command executes a command procedure named SETUP.FIL cataloged in the directory COMMON on the disk DBB2.

For command procedures that you execute frequently, you can define a symbol name as a synonym for the @ command line. For example:

```
$ SETUP := @DBB2:[COMMON]SETUP.FIL
```

5.2.1 Input and Output Devices

When you use the @ command interactively, the system equates the logical device SYS$INPUT with the command procedure file. SYS$OUTPUT and SYS$ERROR remain assigned to your terminal so that output resulting from command or program execution and system messages are displayed at your terminal.

If you want a permanent record of the output from the execution of a command procedure, you can use the /OUTPUT qualifier of the @ command. The /OUTPUT qualifier defines an equivalence name for the logical name SYS$OUTPUT. For example:

```
$ @TESTALL/OUTPUT=TESTALL.LOG
```

If you execute this command, all the data normally displayed on your terminal during the execution of TESTALL.COM is instead written to the disk file named TESTALL.LOG. Error and warning messages are displayed on your terminal as well.

To determine the outcome of the command procedure, you can use the TYPE command to display the file or the PRINT command to print it.

5.2.2 Verification of Command Procedure Execution

Generally, the output from the interactive execution of a command procedure includes:

- Responses and messages from DCL commands
- All data messages displayed by programs that write to SYS$OUTPUT

If you also want to see the lines in the command procedure displayed at the terminal as they are executed, you can issue the SET VERIFY command. You can issue the SET VERIFY command within the command procedure, or at the interactive command level; the command affects

all command procedures you subsequently execute. By default, the system does not display lines in a command procedure that you execute interactively. You can reverse the effect of SET VERIFY with the SET NOVERIFY command.

For example, to display lines in a particular command procedure, you could place the SET VERIFY command at the beginning of the procedure, and place the SET NOVERIFY command at the end of the procedure, as follows:

```
$ SET VERIFY
$ RUN TESTA
$ RUN TESTB
$ RUN TESTC
$ SET NOVERIFY
```

The SET NOVERIFY command at the end of this procedure suppresses verification for subsequent procedures.

5.3 BATCH JOBS

You can submit command procedures for processing as batch jobs. The format of a batch job command stream is identical to the format described for command procedures. A batch job is, in fact, a command procedure. Batch job queues are set up and maintained by the system manager; in most cases, at least one of these queues will be named SYS$BATCH.

5.3.1 Submitting Batch Jobs from the Terminal

If you use the @ command to execute a command procedure interactively, you cannot enter other commands to do other work while the procedure is executing. If you create and use procedures that require lengthy processing time -- for example, the compilation or assembly of large source programs -- you can submit the procedure for execution as a batch job. Once the job is queued, your terminal is free for you to continue interactive work.

The SUBMIT command requests the system to enter a command procedure in the batch job queue. The SUBMIT command assumes current disk and directory defaults, and it assumes the default file type of COM. For example, to execute the command procedure TESTALL.COM in your default disk directory, issue the command:

```
$ SUBMIT TESTALL
  Job 210 entered on queue SYS$BATCH
```

When you submit a job, the system displays a message showing that the job is queued, gives you the job identification number (jobid) it has assigned to the job, and displays the name of the batch job queue on which it entered the job.

You can specify qualifiers with the SUBMIT command to control when the job is actually available for processing, or you can specify a name for the job, overriding the default file name (the file name of the command procedure file). For example:

```
$ SUBMIT/NAME=BATCH25/HOLD  TESTALL
  Job 203 entered on queue SYS$BATCH
```

This SUBMIT command submits the file TESTALL.COM for processing, placing it in a hold status. The system will not execute the job until you specifically release it. The /NAME qualifier assigns a name to the job; the batch job log file will be named BATCH25.LOG.

The system identifies all batch jobs by their job identification numbers (jobids). You can query and change the status of your jobs by referring to these numbers. After you have submitted a batch job, and before it completes execution, you can use the SHOW QUEUE command to determine its place in the queue. For example:

```
$ SHOW QUEUE/BATCH/FULL

* Batch Queue "SYS$BATCH" Joblim=6, Ini=i=4, Swap

 Current Job 201 WILSON          BATCH01   12-JAN-1978 14:15
 Current Job 202 HARRIS          FORCLG    12-JAN-1978 14:20
 Holding Job 203 HIGGINS         BATCH25   12-JAN-1978 14:22
  DBA1:TESTALL.COM;3
```

If you entered the job in the queue with the /HOLD qualifier, you can release it for processing with the SET QUEUE command, as follows:

```
$ SET QUEUE/ENTRY=203/RELEASE  SYS$BATCH
```

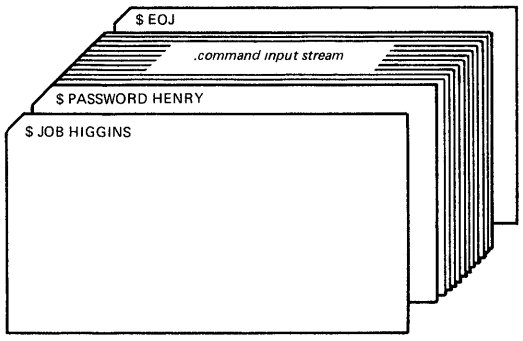
You can also delete it, either before it is processed or while it is executing. Use the DELETE/ENTRY command. For example:

```
$ DELETE/ENTRY=203  SYS$BATCH
```

For details on the SHOW, SET, and DELETE/ENTRY commands, see the command descriptions in Part II.

5.3.2 Submitting Batch Jobs through the Card Reader

When you submit a batch job through the system card reader, you must precede the card deck containing the command procedure with cards containing JOB and PASSWORD commands that specify your user name and password. The last card in the deck should contain the EOJ command. For example:



When the system reads a job from the card reader, it validates the user name and password specified on the JOB and PASSWORD cards. Then, it copies the file into a temporary disk file named INPBATCH.COM in your default directory and queues the job for batch execution.

Thereafter, processing is the same as for jobs submitted interactively with the SUBMIT command. When the batch job completes, the system deletes the temporary file.

When the system reads input from the card reader, it also recognizes two special types of card:

- Translation mode cards
- EOF cards

Translation mode cards in the input stream change the current translation mode. The translation mode is based on the device type of the card punch on which the cards were punched. An 026 punch is indicated by an 026 translation mode card (12-2-4-8 overpunch). An 029 punch is indicated by an 029 translation mode card (12-0-2-4-6-8 overpunch). The default mode can be set with the SET CARD_READER command.

An EOF card (12-11-0-1-6-7-8-9 overpunch) or an EOJ command signals the end of the job. You can also use the EOF card to signal the end of an input stream when you copy a file or read data directly from the card reader.

5.3.3 Input and Output Devices

When the system executes a command procedure submitted to the batch job queue, it creates a detached process to execute the commands. This process receives your disk and directory defaults, and the same resource quotas and privileges assigned to your interactive process. This process is given a name in the format _JOBnnn where nnn is the job identification number assigned to the job. The process executes under your UIC.

The batch job input stream, SYS$INPUT, is equated to the disk containing the command procedure. The output stream, consisting of messages written to SYS$OUTPUT and SYS$ERROR, is equated to a batch job log file. The system catalogs this file in your default disk directory, giving it a file specification of "name.LOG" where name is the file name of the procedure (you can also specify the /NAME qualifier on the JOB command to specify an alternate name). When the batch job completes, the system automatically queues the log file for printing; when the file has printed, the system deletes it.

The batch job log file includes, by default, all command lines executed in the command procedure, system and user-program output to SYS$OUTPUT, and SYS$ERROR, and job termination accounting information.

Note that the batch job log file does not print, and is not deleted, if the job terminates abnormally. For example, if you use the DELETE/ENTRY or STOP command to stop the job, the log file remains in your default directory.

5.4 COMMAND LEVELS

One command procedure can invoke another procedure with the @ command. In this case, the command interpreter reads input from the second file until it reaches the end of the file or until that procedure exits, and then returns to the outer procedure.

The maximum number of command procedures you can nest in this way is eight. For each procedure, the command interpreter redefines the equivalence name for SYS$INPUT, equating it to the file from which the current procedure is read. If the /OUTPUT qualifier is specified with an @ command, the command interpreter redefines the logical name SYS$OUTPUT as well.

The logical names SYS$ERROR, SYS$DISK, and SYS$COMMAND do not change. SYS$COMMAND is always equated to the initial command level: if you execute a procedure interactively, SYS$COMMAND is always equated to your terminal; if you submit a batch job, SYS$COMMAND is always equated to the initial batch input file.

Figure 5-1 illustrates logical name assignments at various command levels.

5.5 THE LOGIN.COM FILE

If you create a file named LOGIN.COM in your default disk directory, the command interpreter automatically executes the procedure each time you log in and at the beginning of each batch job you submit.

Use the LOGIN.COM file to execute any commands or sequences of commands that you normally would want to execute at the start of each terminal session. For example, if you define synonyms for DCL commands, you can place the global assignment statements for the command name synonyms in the LOGIN.COM file so they will be available during every terminal session. For example, a LOGIN.COM file could contain the following statements:

```
$ QP :== SHOW QUEUE/DEVICE
$ QB :== SHOW QUEUE/BATCH
$ TIM :== SHOW TIME
```

Note that these symbols are defined as global symbols, so they will not be deleted when the procedure finishes executing.

A LOGIN.COM file can also contain commands to set up terminal characteristics, assign logical names, run programs, execute command procedures, or display message files.

You can update your LOGIN.COM file at any time to change, add, or delete commands. When you first create the LOGIN.COM file, or after you update it, you can execute it with the @ command so that the commands it contains become effective.

5.6 USING SYMBOLS IN COMMAND PROCEDURES

An assignment statement (= command) equates a character string symbol name with a character string or arithmetic value. You can use assignment statements in command procedures to perform string substitution and manipulation, arithmetic operations, and logical comparisons.

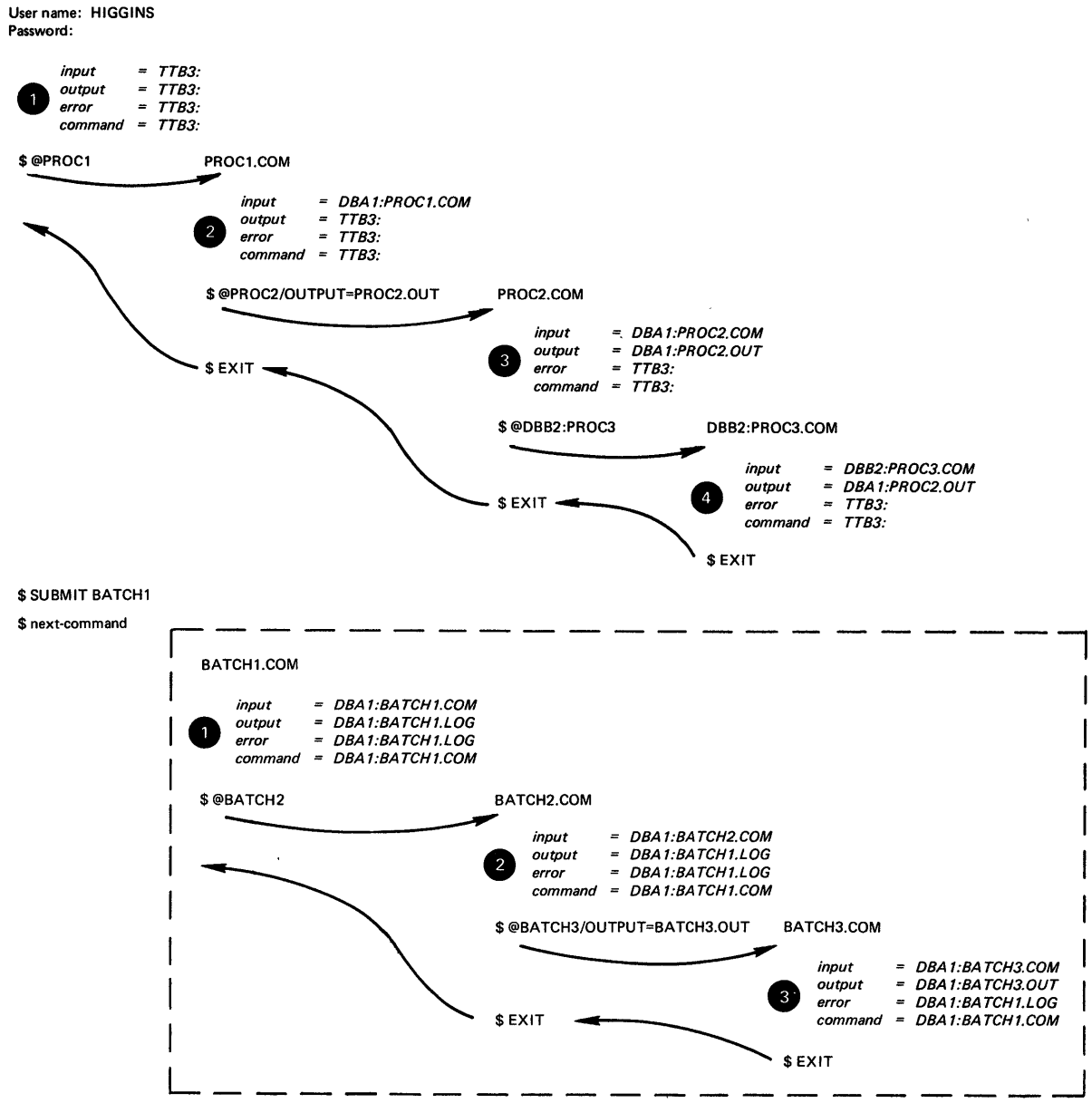


Figure 5-1 Command Levels

5.6.1 Local Symbols

You can define symbols interactively or within command procedures. The command interpreter maintains a symbol table for each active command level, including the interactive command level. These tables are called local symbol tables, and the symbols they list are called local symbols.

To define a symbol as a local symbol, use a single equals sign (=) in an assignment statement. For example:

```
$ COUNT = 1
$ OUTDAT := "Beginning tests...."
```

These assignment statements define local symbols. The first statement equates the symbol named COUNT with a numeric value of 1. The second statement equates the symbol named OUTDAT with a string value; the string is enclosed in quotation marks because it contains literal lowercase letters. A colon (:) in an assignment statement indicates a character string assignment.

A local symbol exists as long as the command level at which it was defined remains active, unless the symbol is specifically deleted. For example, if you define the symbol COUNT interactively, any procedure you subsequently execute can refer to the symbol COUNT and obtain its current value. As another example, the procedure A.COM contains:

```
$ TOTAL = 1
$ @B
```

The procedure B.COM contains the line:

```
$ NEWTOTAL = TOTAL + 1
```

When B executes, the symbol name TOTAL is accessible and can be substituted, because the command level at which procedure A is executing is still active.

Local symbols are deleted when the procedure that defined them finishes executing. In the above example, the symbol NEWTOTAL defined by B is deleted when B completes.

5.6.2 Global Symbols

In addition to the local symbol tables, the command interpreter also maintains a global symbol table. A global symbol exists for the life of the job, unless specifically deleted, and is recognized at any command level. To define a global symbol, use two equals signs (==) in the assignment statement. For example:

```
$ RESULT == 50
$ FILENAME :== MYFILE.DAT
```

These assignment statements define global symbols.

5.6.3 Symbol Substitution

To request symbol substitution, enclose a symbol name in apostrophes, for example:

```
$ TYPE 'FILENAME'
```

You can omit a trailing apostrophe at the end of a command line.

The command interpreter automatically substitutes symbols in the context of an arithmetic expression. For example:

```
$ TOTAL = COUNT + 1
```

No apostrophes are needed to request substitution of the symbol COUNT in this arithmetic assignment statement because the command interpreter automatically substitutes symbols while it executes arithmetic assignments. Note, however, that you must use an apostrophe in a string assignment statement:

```
$ OLDSTRING := 'FILENAME'
```

Otherwise, the command interpreter would equate the symbol named OLDSTRING with the literal string FILENAME, rather than using the current value of FILENAME.

Within character strings enclosed in quotation marks, you can request symbol substitution by preceding a symbol name with two apostrophes. For example:

```
$ PROMPT_STRING := "Creating file ''FILENAME'.TST"
```

Only a single apostrophe is required to delimit the end of the symbol name.

When the command interpreter performs substitution on symbols, it searches symbol tables in the following order:

1. The local symbol table for the current command level
2. Local symbol tables for each previous command level, searching backwards from the current level
3. The global symbol table

You can use the SHOW SYMBOL command to display the current value of any symbol. The SHOW SYMBOL command uses the same order of search to locate symbol definitions, that is, it searches the local symbol tables, then the global symbol table, to locate a specified symbol name.

For complete details on the syntax of assignment statements and symbol substitution, see the description of the Assignment Statement (= command) in Part II. For additional details on arithmetic and logical operations you can perform on symbols and literal values, see Section 6.6, "Rules for Forming Expressions."

5.6.4 Passing Parameters to Command Procedures

You can write generalized command procedures that may perform in a different way each time you execute them. The command interpreter defines eight special symbols for use as parameters within command procedures. These local symbols are named P1, P2, P3, and so on, to P8; they are all initially equated to null strings. You can pass numeric or character string values for these parameters when you execute a procedure with the @ command or submit it as a batch job with the SUBMIT command.

For example, the procedure named RUNTEST contains the lines:

```
$ ASSIGN 'P1' INFILE
$ ASSIGN 'P2' OUTFILE
$ RUN SORTER
```

The program SORTER.EXE reads a file using the logical name INFILE and writes an output file using the logical name OUTFILE. To assign equivalences to these logical names, you must provide values for P1 and P2 when you execute the procedure. If you execute the procedure with the @ command, you could enter the values for P1 and P2 as command parameters, as follows:

```
$ @RUNTEST INSORT.DAT OUTSORT.DAT
```

This @ command line gives the symbol name P1 a value of INSORT.DAT and the symbol P2 a value of OUTSORT.DAT.

To pass parameters with the SUBMIT command, use the /PARAMETERS qualifier, as follows:

```
$ SUBMIT RUNTEST/PARAMETERS=(INSORT.DAT,OUTSORT.DAT)
```

When you specify more than one parameter, separate them with commas and enclose them in parentheses as in the above example.

Note that in both of the above examples the values for the parameters are assigned according to the order in which you specify them, that is, the first parameter you enter is P1, the second is P2, and so on.

Figure 5-2 illustrates a command procedure testing the parameters that are passed to it. The procedure establishes a symbol named COUNT and concatenates the value of COUNT with the letter P to test each parameter. If an invalid parameter is passed, the procedure constructs an error message. If this procedure, named MASTER.COM, is invoked with the line:

```
$ @MASTER PURGE RENAME
```

It performs its main function, that is, concatenates a group of files into a single file, and it performs its PURGE option. However, when the value of COUNT reaches 2, it checks the value of P2 and determines that it is not a valid option. Then, it uses a WRITE command to issue the error message:

```
Bad parameter "RENAME"
```

The WRITE command is described later in this chapter, in Section 5.8, "Reading and Writing Files."

```
$ SET NOVERIFY          ! Ensure that verification is off
$ !
$ ! Concatenates all TXT files into a single master file named MASTER.DOC
$ ! and optionally prints the file, purges earlier versions, and changes
$ ! the file protection code.
$ !
$ ! First, concatenate all existing files into MASTER
$ !
$ COPY/LOG *.TXT/CONCATENATE MASTER.DOC
$ !
$ ! Loop to check for keyword parameters PRINT, PROTECT, PURGE
$ ! Parameters can be entered in any order. The first
$ ! command in the loop checks for a null; when there are no
$ ! more parameters, the procedure exits.
$ !
$ COUNT = 0
$ LOOP:
$ COUNT = COUNT + 1
$ IF P'COUNT.EQS."" THEN EXIT
$ !
$ !
$ ! Use logical AND operation to test current parameter; if it is not
$ ! any of the valid keywords, issue an error message and continue looping.
$ !
$ IF P'COUNT.NES."PRINT".AND.P'COUNT.NES."PURGE".AND. -
  P'COUNT.NES."PROTECT" THEN GOTO BADPARAM
$ !
$ ! Use IF commands to check the current parameter against each keyword and
$ ! provide an appropriate branch.
$ !
$ IF P'COUNT.EQS."PRINT" THEN  GOTO PRINT
$ !
$ IF P'COUNT.EQS."PROTECT" THEN GOTO PROTECT
$ !
$ IF P'COUNT.EQS."PURGE" THEN GOTO PURGE
$ !
$ ! If any parameter is PRINT, print the document file
$ !
$ PRINT:
$ PRINT/COPIES=20/AFTER=17:00 -
  MASTER.DOC/NAME=DOCUMENT_REVIEW
$ GOTO LOOP             ! Go check for more parameters
$ !
$ ! If any parameter is PROTECT, change file protection
$ !
$ PROTECT:
$ SET PROTECTION MASTER.DOC/PROTECTION=(GROUP:RWED)
$ GOTO LOOP             ! Go check for more parameters
$ !
$ ! If any parameter is PURGE then purge old  files
$ !
$ PURGE:
$ PURGE *.TXT,MASTER.DOC
$ GOTO LOOP             ! Go check for more parameters
$ !
$ BADPARAM:             ! Construct error message
$ PARM := P'COUNT'
$ WRITE SYS$OUTPUT "Bad parameter ",""""",'PARM',""""
$ GOTO LOOP
```

Figure 5-2 Testing Parameters Passed to a Command Procedure

5.6.5 The INQUIRE Command

When you execute a command procedure interactively, you can use the INQUIRE command to define a value for a symbol while the procedure executes. When the INQUIRE command is executed, the command interpreter issues a prompting message to SYS$COMMAND, that is, the terminal; the text of the message is taken from the INQUIRE command line, as shown in the example below.

The procedure RUNTEST contains the lines:

```
$ INQUIRE IN INPUT FILE
$ INQUIRE OUT OUTPUT FILE
$ ASSIGN 'IN' INFILE
$ ASSIGN 'OUT' OUTFILE
$ RUN SORTER
```

When you execute this procedure, the terminal interaction might appear as follows (if SET NOVERIFY is in effect):

```
$ @RUNTEST
  INPUT FILE: DB1:INSORT.DAT
  OUTPUT FILE: DB2:OUTSORT.DAT
$
```

When these INQUIRE commands are executed, the prompting messages INPUT FILE and OUTPUT FILE are displayed, and you must enter values for the symbols IN and OUT before the command procedure continues. The prompting strings INPUT FILE and OUTPUT FILE are optional parameters for the INQUIRE command; if you do not specify them, the command uses the symbol names IN and OUT to prompt for values.

5.6.6 Deleting Symbols

The DELETE/SYMBOL command deletes a local symbol from the local symbol table for the current command level and global symbols. For example, the following command deletes the local symbol named TOTAL:

```
$ DELETE/SYMBOL TOTAL
```

The /GLOBAL qualifier indicates a global symbol is to be deleted. For example:

```
$ DELETE/SYMBOL/GLOBAL/ALL
```

This command deletes all global symbols.

5.7 COMMANDS TO CONTROL THE EXECUTION OF A COMMAND PROCEDURE

Normally, the command interpreter executes each command in a command procedure in sequential order, and terminates processing when it reaches the end of the command procedure file. However, using combinations of the following commands, you can control the sequence in which lines are executed; conditionally execute lines; construct loops; and handle errors in command procedures:

```
IF      EXIT
GOTO    STOP
ON
```

Some techniques for using each of these commands are shown in the following sections. For additional examples, see the individual command descriptions in Part II.

5.7.1 The IF Command

The IF command tests a symbol value or an arithmetic expression and executes a given command if the result of the given expression is true. The following examples show valid expressions used in IF commands.

Example	Explanation
`$ IF A + B .EQ. 10 THEN command`	Executes the given command if the sum of the defined symbols A and B is 10
`$ IF A THEN command`	Executes the given command if the symbol A has an odd numeric value or is equated to a character string that begins with the letter Y (yes) or the letter T (true)
`$ IF .NOT. A THEN command`	Executes the given command if the symbol A has an even numeric value or is equated to a character string that begins with the letter N (no) or the letter F (false)
`$ IF COUNT.LE.100 THEN command`	Executes the given command if the symbol COUNT has a current value less than or equal to 100
`$ IF P1.EQS."TYPE" THEN command`	Executes the given command if the first parameter passed to the command procedure was the word TYPE

The target command of an IF command can be any valid DCL command; you can optionally precede the command with a dollar sign. If the specified expression is true, the command interpreter processes the command: it substitutes symbol names, if any, with their current values; checks the command qualifiers and parameters for validity; and executes the command. If the expression is false, the command interpreter does not process the command but continues processing with the next line in the file.

Note that any symbols you use in expressions in the IF command must be previously defined; if a symbol is undefined, the IF command issues a warning message and does not execute the target command.

For a complete list of the operations you can perform in arithmetic expressions, see the description of the IF command in Part II. See Section 6.6 "Rules for Forming Expressions" for additional information on the syntax of expressions.

5.7.2 The GOTO Command

The GOTO command passes control to a labeled line in a command procedure. Labels must appear as the first item on a line and must be terminated with a colon (:). For example:

```
$ GOTO BYPASS
   .
   .
$ BYPASS:
```

The GOTO command is especially useful as the target command of an IF command: you can cause a procedure to branch forward or backward according to variable conditions, or according to parameters that you pass to the procedure.

For example, when you use parameters in a command procedure, it is good practice to test the parameters at the beginning of the procedure. A procedure that you execute interactively could begin with the lines:

```
$ IF P1.NES."" THEN GOTO OKAY
$ INQUIRE P1 ENTER FILE SPEC
$ OKAY:
$ PRINT/HOLD/COPIES=10/FORMS=B  'P1'
```

In this example, the IF command checks that P1 is not a null string (remember, the command interpreter initially defines P1 through P8 as null strings). If P1 is a null string, the GOTO command is not executed and the INQUIRE command prompts you to enter a value for the parameter. Otherwise, the GOTO command causes a branch around the INQUIRE command. In either case, the procedure executes the PRINT command following the line labeled OKAY.

5.7.3 The EXIT and STOP Commands

The EXIT command terminates execution of the current command procedure, and returns control to the calling command level. If you execute the procedure interactively, the EXIT command returns control to the interactive level; if you execute the procedure from within another procedure, control returns to the nextmost outer procedure.

The EXIT command is useful for writing procedures that have more than one execution path. For example:

```
$ START:
$ IF P1.EQS."TAPE".OR.P1.EQS."DISK" THEN GOTO 'P1'
$ INQUIRE P1 ENTER DEVICE (TAPE OR DISK)
$ GOTO START
$ TAPE:  !  PROCESS TAPE FILES
    .
    .
$ EXIT
$ DISK:  !  PROCESS DISK FILES
    .
    .
$ EXIT
```

To execute this command procedure, you must enter either TAPE or DISK as a parameter. The IF command uses a logical OR to test whether either of these strings was entered. If so, the GOTO command branches

appropriately, using the parameter as the branch label. If P1 was neither TAPE nor DISK, the INQUIRE command prompts for a correct parameter; the GOTO START command establishes a loop.

The commands following each of the labels TAPE and DISK provide different paths through the procedure. The EXIT command following the commands after the label TAPE ensures that after these commands are processed, the commands after the label DISK are not executed.

Note that the EXIT command at the end of the procedure is not required because the end-of-file of the procedure causes an implicit EXIT command.

The STOP command also terminates execution of a procedure; however, when a STOP command is executed in a nested command procedure, the outer procedure(s) are also terminated. If you execute the procedure interactively, control returns to the interactive command level; if you execute the procedure in a batch job, the batch job terminates.

5.7.4 Testing Status Values

When a DCL command, user program, or command procedure completes execution, the command interpreter saves the completion status value in a reserved global symbol named $STATUS. For example, if an error occurs following a TYPE command, the value in $STATUS represents the error code returned by the TYPE command. When a command or program completes successfully, $STATUS has a value of %X00000001. Note that the command interpreter always maintains the current value of $STATUS in hexadecimal.

You cannot explicitly set a value for $STATUS with an assignment statement, but you can set a value for $STATUS on an EXIT command that returns control to a previous command level. For example, suppose the procedure A.COM contains:

```
$ @B
$ IF $STATUS .EQ.  2 THEN GOTO CONTROL
   .
   .
```

The procedure B.COM can contain the line:

```
$ EXIT 2
```

This EXIT command places the value 2 in the global symbol $STATUS, which is tested by the calling procedure, A.COM.

5.7.4.1 **Severity Levels** - The low-order three bits of the status value contained in $STATUS represents the severity of the completion status. The reserved global symbol $SEVERITY always contains only this portion of the status value. These values, and the severity levels they represent are:

Value	Severity
0	Warning
1	Success
2	Error
3	Information
4	Severe, or fatal, error

Note that since successful codes have odd numeric values and warning and error codes have even values, you can test for the successful completion of a command or program with IF commands testing either value, as in the following examples:

```
$ IF $SEVERITY THEN ...
$ IF $STATUS THEN ...
```

If the current value in $SEVERITY or $STATUS is odd, then the command or program completed successfully and the IF expressions are true. Otherwise, the IF expressions are false.

5.7.4.2 **The ON command** - During the execution of a command procedure, the command interpreter checks the return status from each command or program that executes. With the ON command, you can establish a course of action for the command interpreter based on the result of the check.

By default, the command interpreter executes an EXIT command when an error or severe error occurs, and continues when warnings occur. You can override this default with the ON command. For example, if you want a procedure to exit when warnings occur, use the command:

```
$ ON WARNING THEN EXIT
```

If you want the procedure to continue if a warning or an error occurs, use the command:

```
$ ON ERROR THEN CONTINUE
```

An ON command action is executed only once; thus, if you have used the above command, the command interpreter continues after an error occurs, but resets the default condition. If a second error occurs, and no other ON command is executed, the procedure exits.

If you nest command procedures, you can establish a different ON condition for each procedure. An ON condition in an outer procedure does not affect the execution of a nested procedure.

You can request the command interpreter to not check the status returned from any commands with the SET NOON (that is, no ON) command. For example:

```
$ SET NOON
    .
    .
$ SET ON
```

In the above example, the SET NOON command requests the command interpreter to perform no action if any type of error occurs. The SET ON command restores the current ON condition action; that is, whatever condition was in effect before the SET NOON command was executed.

5.7.4.3 **System Status Values** - When control returns to the DCL command level, the command interpreter always tests the current value of $STATUS. If it contains an even numeric value, and if the high-order digit is 0, the command interpreter displays the system warning or error message associated with that status code.

Note that normally when a command procedure exits following a warning or error condition, the command interpreter sets the high-order digit of $STATUS to 1, leaving the remainder of the value intact. Many system programs that issue their own messages set this field to 1 also, so that the command interpreter does not redisplay the messages. However, if a command procedure contains an EXIT command that explicitly uses the current value of $STATUS as the exit status code, the command interpreter does display the message associated with the status code.

For example, a command procedure can contain the line:

```
$ ON WARNING THEN EXIT '$STATUS'
```

When any warning, error, or severe error occurs during the execution of this procedure, the procedure exits with the unchanged status value. Because any of these conditions normally cause system messages, the command interpreter will display the message associated with the status value.

Most DCL commands invoke system utilities that generate unique status values and error messages based on different results. Please note the following exceptions:

1. The commands listed below, on successful completion, do not change the current value of $STATUS or $SEVERITY:

CONTINUE	GOTO
DEPOSIT	HELP
EOD	IF
EXAMINE	SHOW
EXIT	STOP
	WAIT

   If any of these commands results in a nonsuccessful status, however, that status value is placed in $STATUS, and the severity level is placed in $SEVERITY.

2. The commands listed below set only the severity level portion of $STATUS; they also change the value of $SEVERITY:

COBOL/RSX11	LINK/RSX11
DIFFERENCES	MACRO
DUMP	
EDIT	

3. The commands listed below always set the value of $STATUS and $SEVERITY to a successful code:

CREATE	RENAME
DIRECTORY	SET PROTECTION
LIBRARY	SORT/RSX11
	UNLOCK

The programs invoked by the commands listed in (2) and (3) above issue their own error messages. The message code associated with the status values returned by these commands is EXITSTATUS.

5.7.5 The SYNCHRONIZE and WAIT Commands

The SYNCHRONIZE and WAIT commands both place a job in a wait state: the SYNCHRONIZE command waits for the completion of another job; the WAIT command waits for a specified period of time to elapse. For example, if jobs are submitted concurrently to perform cooperative functions, one job can contain the command:

```
$ SYNCHRONIZE BATCH25
```

After this command is executed, the procedure cannot continue execution until the job identified by the job name BATCH25 completes.

The WAIT command is useful for procedures that must have access to a shared system resource, for example a disk or tape drive. The following example shows a procedure that requests the allocation of a tape drive; if the command does not complete successfully, the procedure places itself in a wait state, and then retrys the request:

```
$ TRY:
$ ALLOCATE DM:  RK:
$ IF $STATUS THEN GOTO OKAY
$ WAIT 00:05
$ GOTO TRY
$ OKAY:
$ REQUEST/REPLY/TO=DISKS - .
"Please mount BACK_UP_GMB on ''F$LOGICAL("RK")' "
   .
   .
```

The IF command following the ALLOCATE request checks the value of $STATUS. If successful, the procedure continues. Otherwise, it issues the WAIT command; the WAIT command specifies a time interval of 5 minutes. After waiting 5 minutes, the next command, GOTO, is executed, and the request is repeated. This procedure continues looping and attempting to allocate a device until it succeeds, or until the batch job is deleted or stopped.

The REQUEST command in the above example illustrates the F$LOGICAL lexical function, which translates the logical name RK assigned to the allocated device. Lexical functions are described in Section 5.9.

5.8 READING AND WRITING FILES

With the DCL command interpreter, you can read, write, and create files. The basic steps to read and write files are:

1. Use the OPEN command to open a file. The OPEN command assigns a logical name to the file, and specifies whether the file is to be read or written. A file cannot be both read and written at the same time.

2. Use the READ or WRITE command to read or write the file. The READ and WRITE commands use command symbols as buffers to contain input and output records; the READ command reads a record into a symbol and the WRITE command writes one or more symbols or literal character strings into a single record in an output file.

3. Use the CLOSE command to close the file. After a file has been opened with the OPEN command, it remains open until specifically closed.

The following example shows a procedure that reads an input file and copies each record into an output file.

```
$ OPEN/READ INFILE DATA.TST
$ OPEN/WRITE OUTFILE DATA.OUT
$ READLOOP:
$ READ/END_OF_FILE=FINISH INFILE RECORD
$ WRITE OUTFILE RECORD
$ GOTO READLOOP
$ FINISH:
$ CLOSE INFILE
$ CLOSE OUTFILE
```

The OPEN commands open the files DATA.TST and DATA.OUT for input and output, respectively. The READ command specifies the /END_OF_FILE qualifier so that the GOTO command that causes a loop is skipped when the end-of-file is reached on DATA.TST. Each READ command changes the value of the symbol RECORD; the WRITE command writes the record into the file DATA.OUT.

When all records have been read, the procedure goes to the label FINISH, where the CLOSE commands close the files.

5.8.1 Reading and Writing Process Permanent Files

You can also use the READ and WRITE commands to read data from the current input device or to write messages on the current output device. The process permanent files SYS$INPUT, SYS$OUTPUT, SYS$COMMAND, and SYS$ERROR do not have to be explicitly opened before you refer to them in READ or WRITE commands. For example:

```
$ READ SYS$COMMAND TESTID
```

This READ command results in a read to the current device SYS$COMMAND; thus, when the procedure is executed interactively, the read is issued to the terminal. When the READ command executes, the command interpreter displays the following prompt at the terminal:

```
Data:
```

Whatever you type in response to this prompt is then equated to the symbol named TESTID.

Similarly, you can write a line of data to the terminal, or whatever the current output device is, with the WRITE command. For example:

```
$ WRITE SYS$OUTPUT "Count is ''COUNT'... continuing..."
```

Before this line is displayed on the terminal, the symbol named COUNT is substituted with its current value.

5.8.2 File Formats

When you create a file with the WRITE command, you cannot specify any attributes for the file: the command interpreter always creates a file in print file format, with carriage control bytes preceding and following each record.

These files are therefore not compatible with files created by the default system editor SOS, or RSX-11M utilities invoked by DCL commands, for example, SORT-11 (invoked by the SORT/RSX11 command). If you create a file with the DCL command WRITE and you wish to use the file as input to another program or command, you can perform an intermediate step to convert the file to a suitable format. One simple way to do this to invoke the SOS editor to edit the file and then write the file back onto disk. SOS removes the carriage control bytes from each record as it writes the output file. For example:

```
$ OPEN/WRITE OUTFILE DATA.OUT
    .
    .
$ CLOSE OUTFILE
$ EDIT/NOLINES DATA.OUT
EB
```

After a file is closed, you can specify it as an input file to the editor: the /NOLINES qualifier in this example indicates to SOS that the file does not have line numbers associated with the records in the file. The SOS command EB follows the EDIT command in the input stream for the procedure: the EB command writes the file onto disk without incrementing the version number.

5.9 LEXICAL FUNCTIONS

The command interpreter recognizes a set of functions, called lexical functions, that return information about character strings and attributes of the current process.

You can use lexical functions in any context in which you normally use symbols or expressions. In command procedures, you can use lexical functions to translate logical names, perform character string manipulations, and determine the current processing mode of the procedure.

Table 5-1 summarizes the valid functions, their formats, and the information returned by each. Some examples of using lexical functions are given in the next few subsections. The syntax requirements for specifying lexical functions are described in detail in Section 6.7 "Rules for Specifying Lexical Functions."

Table 5-1
Summary of Lexical Functions

Function	Value Returned
F$CVSI(bit-position,count,integer)	Signed value extracted from the specified integer, converted to an ASCII literal
F$CVUI(bit-position,count,integer)	Unsigned value extracted from the specified integer, converted to an ASCII literal

(continued on next page)

Table 5-1 (Cont.)
Summary of Lexical Functions

Function	Value Returned
F$DIRECTORY()	Current default directory name string, including brackets
F$EXTRACT(offset,length,string)	Substring beginning at specified offset for length specified of indicated string
F$LENGTH(string)	Length of specified string
F$LOCATE(substring,string)	Relative offset of specified substring within string indicated; or, the length of the string if the substring is not found
F$LOGICAL(logical-name)	Equivalence name of specified logical name (first match found in ordered search of process, group, and system logical name tables); or, a null string if no match is found
F$MESSAGE(code)	Message text associated with the specified numeric status code value
F$MODE()	One of the character strings INTERACTIVE or BATCH
F$PROCESS()	Current process name string
F$TIME()	Current date and time of day, in the format dd-mmm-yyyy hh:mm:ss.cc
F$USER()	Current user identification code (UIC), in the format [g,m]
F$VERIFY()	A numeric value of 1 if verification is set on; a numeric value of 0 if verification is set off

5.9.1 F$MODE and F$VERIFY

The F$MODE function is useful in procedures that must act differently when executed in batch mode and interactively.

For example, you could create a LOGIN.COM file that tests whether the procedure is being executed as a result of an interactive login or at the beginning of a batch job by using the F$MODE function, as shown in the following example:

```
$ IF "'F$MODE()'" .EQS.  "BATCH" THEN GOTO BATDEF
$ INTDEF:
    .
    .
$ EXIT
$ BATDEF:
    .
    .
```

All function names must be preceded with the substitution operator (') and must include parentheses as argument delimiters even when the function requires no arguments, as in the above example.

Note that the function name F$MODE in the above example is enclosed in quotation marks, and that it is preceded by two apostrophes. Lexical functions are performed before the command string is parsed. Thus, when a lexical function is used in an expression which normally causes symbol values to be substituted, and the output of the function is being used as a literal character string, the function must be enclosed in quotation marks. Because the function is used within quotation marks, two apostrophes are required.

The F$VERIFY function returns a value of 0 or 1, based on whether verification of command procedures is off (0) or on (1). You can use this function to test or to save the current setting. For example, a procedure can save the current setting before changing it and then later restore the setting:

```
$ SAVE_VERIFY = 'F$VERIFY()'
$ SET NOVERIFY
    .
    .
$ IF SAVE_VERIFY THEN SET VERIFY
```

The assignment statement saves the current setting of verification before the SET NOVERIFY command sets verification off. Later, the value of SAVE_VERIFY is tested; if it is true, that is, if it has a value of 1, it indicates that verification was previously on. Otherwise, verification was initially off and remains off.

5.9.2 F$EXTRACT, F$LOCATE, F$LENGTH, and F$TIME

The F$EXTRACT, F$LOCATE, and F$LENGTH functions allow you to manipulate character strings. The following example shows a procedure named TYPE.COM that accepts a file specification as a parameter. It uses the F$LOCATE function to determine whether the file specification contains a period (.), indicating that a file type is present. If not, it provides a default file type defined by the symbol FTYPE.

```
$ FTYPE := .TXT
$ IF 'F$LOCATE(".",P1)' .EQ. 'F$LENGTH(P1)' -
  THEN P1 := 'P1'FTYPE
$ TYPE 'P1
```

In this example, the F$LOCATE function locates a period; if no period is present, the function returns the length of the string. The function returned is compared with the function F$LENGTH which always returns the length of the string. If this command procedure is invoked as shown below, it types the file ALPHA.TXT:

```
$ @TYPE ALPHA
```

Note that this example assumes that the file specification does not contain a directory string.

The F$TIME function returns the current date and time. The following example shows a procedure that checks the current time of day by locating and extracting the hours from the date and time string returned by F$TIME:

```
$ TIME := 'F$TIME()'
$ HOUR = 'F$LOCATE(" ",TIME)' + 1
$ HOUR_LENGTH = 'F$LOCATE(":",TIME)' - HOUR
$ IF 'F$EXTRACT(HOUR,HOUR_LENGTH,TIME)' .GT. 12 THEN GOTO LUNCH
$ MORNING:
$ WRITE SYS$OUTPUT "Not yet, sigh...."
$ EXIT
$ LUNCH:
$ WRITE SYS$OUTPUT "Go to lunch..."
$ EXIT
```

The string returned by F$TIME always contains one blank between the date and time field. The F$LOCATE function locates the hours field of the date and time string by adding 1 to the location of the blank. The next assignment statement determines the end of the hours field by locating the colon (:) that delimits the hours from the seconds, then subtracts the value of the beginning of the field (in the symbol HOURS) from the location of the colon to determine the length of the hours field. Last, the current hour is extracted from the string by using the values obtained from the F$LOCATE functions, and the result is compared with the number 12.

Note that all arguments specified for lexical functions are candidates for automatic substitution. Thus, the symbols HOUR, TIME, and so on, are substituted before the function is performed. All literal characters or character strings in functions must be placed in quotation marks.

5.9.3 F$DIRECTORY, F$PROCESS, and F$USER

The F$DIRECTORY, F$PROCESS, and F$USER functions return information about the current process: the current default directory, the process name, and the UIC. These functions are all returned as character strings, so you must enclose the functions in quotation marks if you want to use them as literal character strings in expressions. For example:

```
$ IF "''F$DIRECTORY()'" .EQS. "['F$PROCESS()']" THEN ...
```

This IF command tests whether the current default directory string is the same as the current process name. You could also test whether the current default directory is the same as the current UIC, as follows:

```
$ IF "''F$DIRECTORY()'" .EQS. "''F$UIC()'" THEN ...
```

You can also use the F$DIRECTORY function to save the current default directory:

```
$ SAVE_DIR := 'F$DIRECTORY()'
$ SET DEFAULT [MALCOLM.TESTFILES]
    .
    .
$ SET DEFAULT 'SAVE_DIR'
```

In this example, the assignment statement equates the current directory to the symbol SAVE_DIR before the SET DEFAULT command establishes a new default directory. Later, the symbol SAVE_DIR is used in the SET DEFAULT command that restores the original default directory.

In this example, quotation marks are not required around the lexical function F$DIRECTORY because these commands do not automatically evaluate expressions, but accept character string data literally.

5.9.4 F$LOGICAL

The F$LOGICAL function translates a logical name and returns the equivalence name string; the translation is not recursive. For example, you can use the F$LOGICAL function to save the current equivalence of a logical name and later restore it. The following example shows how the technique for saving a directory string shown in the preceding section can be amplified to save the default disk device as well:

```
$ SAVE_DIR := 'F$LOGICAL("SYS$DISK")''F$DIRECTORY()'
```

This assignment statement concatenates the results of two lexical functions. Note that the logical name SYS$DISK is enclosed in quotation marks. The command interpreter assumes that any character string specified as an argument to a lexical function that is not enclosed in quotation marks is a symbol; if the symbol is undefined, the command interpreter replaces it with a null string.

CHAPTER 6

GRAMMAR RULES

This chapter describes the syntax rules for the VAX/VMS command language, including rules for:

- Entering commands
- Entering file specifications
- Entering qualifiers
- Entering character string data
- Entering numeric values
- Forming expressions
- Specifying lexical functions
- Entering date and time values

6.1 RULES FOR ENTERING COMMANDS

A command string is the complete specification of a command, including the command name, command qualifiers, parameters, and file qualifiers, if any.

The general format of a command is:

```
command name[/qualifiers...] parameter[/qualifiers...][...]
```

Because you can continue a command on more than one line, the term command string is used to define the entire command that is passed to the system. You can optionally precede any command line with a dollar sign ($) character. The command interpreter ignores the dollar sign.

Each item in a command must be properly delimited, as follows:

- At least one blank character must separate the command name from the first parameter, and at least one blank must separate each additional parameter from the previous parameter. Multiple spaces and tabs are permitted in all cases where a single blank is required.
- Each qualifier must be preceded with a slash (/). The slash can be preceded or followed with none, one, or more blanks, spaces, or tabs.

- If a label precedes the command, the label must be terminated with a colon (:). The colon can be preceded or followed with none, one, or more blanks, spaces, or tabs.

In addition, any special characters you enter on a command can be treated as delimiters, depending on the context of the command. These characters are listed in Table 6-1.

6.1.1 Rules for Continuing Commands on More than One Line

You can enter a command string on more than one line by using the continuation character, a hyphen (-), as the last element on a command line.

Command line continuation is especially useful when you enter a command and want to specify many qualifiers, or when you place a command in a command procedure file and want to make the procedure more readable. For example:

```
$ PRINT MYFILE -
 /AFTER=17:00 -
 /COPIES=20 -
 /NAME="COMGUIDE"
```

Note that when you continue a command, you must still provide the required space before each parameter.

There is no restriction on the number of continued lines you can use to enter a command. However, the maximum number of characters that you can enter in any entire command string is approximately 500.

NOTE

Some DCL commands invoke RSX-11M utility programs. These commands are actually "back translated" to form command strings for the 11-M utility. The maximum length of these lines, after they have been back translated, is 80 characters.

6.1.2 Rules for Entering Comments

Indicate a comment by preceding the comment with an exclamation mark (!). Comments are valid:

- As the first item on a command line; in this case, the entire line is considered a comment and is not processed by the command interpreter
- Following the last character in a command string, or after a hyphen in a command line

Some examples of valid placement of comments follow:

```
$ !THIS ENTIRE LINE IS A COMMENT
$ PRINT MYFILE - !  PRINT COMMAND COMMENT
$_/COPIES=3    !  3 COPIES, PLEASE
```

When you enter a command interactively and continue the command on more than one line, the command interpreter uses the $_ prompt to indicate that it is processing a command.

6.1.3 Rules for Truncating Keywords

All keywords that you enter as command input can be abbreviated by truncating letters on the right. You can truncate:

- Command names
- Command keyword parameters
- Qualifiers
- Qualifier keyword values

All command name keywords are unique when truncated to their first four characters. All keywords recognized by individual commands are similarly unique with respect to other keywords recognized by the same command. Any keyword can be truncated to fewer than four characters as long as the truncation is unique. In fact, when the command interpreter reads a command line, it only examines the first four characters of each keyword you type.

For example, the DEALLOCATE and DEASSIGN commands have the same first three letters. These commands can be truncated to a minimum of four letters.

The TYPE command, however, is the only command that begins with the letter "T." Therefore, the TYPE command has a minimum truncation of one letter.

When you type commands in command procedure files, you should always use at least a four-character truncation, to ensure the compatibility of your command procedure for future releases of the system. It is recommended that in command procedures you always use the full command name, for readability.

Exceptions: The following commands are exceptions to the minimum truncation rule. You can truncate any of these commands to its first letter:

```
CONTINUE
DEPOSIT
EXAMINE
RUN
```

6.2 RULES FOR ENTERING FILE SPECIFICATIONS

The format of VAX/VMS file specifications is described in detail in Chapter 2, "File Specifications and Logical Names." Use this format for all files you specify as parameters to DCL commands. Remember that all file specifications that you enter as parameters to system commands or as qualifier values can be subject to:

- Logical name translation
- The application of defaults

Some commands perform only a single level of logical name translation; that is, translation is not recursive. When this is the case, the parameter description indicates that fact.

In many cases, a command applies a unique default file type to input or output file specifications. The parameter descriptions indicate the default file type, if any.

The parameter descriptions also indicate whether you can specify wild cards in a field in the file specification.

6.2.1 Rules for Entering File Specification Lists

An input file parameter for many commands has the format:

```
file-spec,...
```

This format indicates that you can enter more than one file specification. You can separate the file specifications with commas (,) or plus signs (+). None, one, or more blanks or tab characters can precede or follow the commas or plus signs. The list of file specifications is treated as a single parameter; the system applies temporary defaults to the files specified in the list.

The parameter description always states how the command interprets the list: in most cases, commas and plus signs are equivalent.

6.3 RULES FOR ENTERING QUALIFIERS

Commands can take one or both of the following types of qualifier:

- Command qualifiers
- File qualifiers

Command qualifiers have the same meaning regardless of whether they appear following the command name or following a command parameter. For example:

```
$ PRINT/HOLD SPRING.SUM,FALL.SUM
$ PRINT SPRING.SUM,FALL.SUM/HOLD
```

The /HOLD qualifier is a command qualifier; therefore, the two PRINT commands shown above are equivalent. Both files are placed in a hold status.

File qualifiers, however, sometimes have different meanings depending on where you place them in the command. If specified immediately after a file specification parameter, they affect only the file they follow. If specified after the command name, they affect all files specified as parameters.

For example:

```
$ PRINT/COPIES=2  SPRING.SUM,FALL.SUM
$ PRINT  SPRING.SUM/COPIES=2,FALL.SUM
```

The first PRINT command shown above requests two copies of each of the files SPRING.SUM and FALL.SUM. The second PRINT command requests two copies of the file SPRING.SUM, but only one copy of FALL.SUM.

Some file qualifiers can be applied only to the specification of a parameter and cannot be specified following the command name. When this is the case, the qualifier description indicates that the qualifier is associated with a file parameter. In all other cases, if you specify a file qualifier following the command name, the qualifier is applied to all files specified (if you specify more than one).

6.3.1 Rules for Determining Qualifier Defaults

Qualifiers fall in the following general categories:

- Qualifiers with simple positive and negative forms, for example:

  ```
  /DELETE
  /NODELETE
  ```

  These qualifiers are listed in the command format boxes in Part II, with their default value, as follows:

  ```
  /[NO]DELETE                /NODELETE
  ```

- Qualifiers that require a numeric or character string variable or keyword value. For example:

  ```
  /COPIES=n
  ```

  The default values for these qualifiers are shown in Part 2 as:

  ```
  /COPIES=n          /COPIES=1
  ```

- Qualifiers that accept a file specification value and that also have a negative form, for example:

  ```
  /DEBUG
  /NODEBUG
  /DEBUG[file-spec]
  ```

  These qualifiers are listed in the command format boxes in Part II as follows:

  ```
  /[NO]DEBUG[=file-spec]          /NODEBUG
  ```

  Defaults are provided, where possible.

  NOTE

  Certain qualifiers that request output files accept file-specification values and apply defaults for these specifications in a special way, as described in Section 6.3.3.

- Qualifiers that are overridden by other qualifiers, for example:

  ```
  /PROCESS
  /GROUP
  /SYSTEM
  ```

The default action is shown in Part II with each qualifier in the list, as follows:

```
/GROUP          /PROCESS
/PROCESS        /PROCESS
/SYSTEM         /PROCESS
```

- Qualifiers that affect command execution only if explicitly present and have no corresponding default, for example:

  /RSX

  No defaults are given for this type of qualifier.

When you enter a command, if you specify the same qualifier more than once, both a positive and negative form of the same qualifier, or qualifiers that override one another, the command interpreter accepts only the last specification. For example:

```
$ PRINT MYFILE /COPIES=3/BURST/COPIES=2/NOBURST
```

For this PRINT command, the command accepts only the /COPIES=2 and the /NOBURST qualifiers.

If you enter conflicting qualifiers for a command, the command interpreter generally issues an error message. If a qualifier conflicts with another qualifier, the descriptions of these qualifiers indicate the conflict.

6.3.2 Rules for Entering Qualifier Values

Many qualifiers accept keywords, file specifications, character strings, or numeric values. For keywords, follow the rules for truncating keywords (Section 6.1.3); for other types of value, follow the rules for entering file specifications (Section 6.2), character data (Section 6.4), or numeric values (Section 6.5).

You must separate qualifiers and their values either with equal signs (=) or colons (:). For example, the following specifications are equivalent:

```
/OUTPUT=DBA1:NEW.DAT        /OUTPUT:DBA1:NEW.DAT
```

Many qualifiers accept one or more keyword or variable values. The syntax is represented in the format as:

```
/qualifier=value
```

For example:

```
/COPIES=3
/OVERRIDE=EXPIRATION
```

When more than one value is accepted, the syntax is:

```
/qualifier=value,...
```

If you want to specify more than one value for a qualifier, you must separate the values with commas and enclose them in parentheses. For example:

```
/ENTRY=(1,2,3)
/PARAMETERS=(3,"CYGNUS,LYRA",PRINT)
```

In the second example, the second parameter is enclosed in quotation marks because it contains an embedded comma.

Some qualifier keyword values require additional data. Separate the keyword from its data with a colon or an equals sign. For example:

```
/PROTECTION=GR:RW
/PROTECTION:GR:RW
/PROTECTION=GR=RW
/PROTECTION:GR=RW
```

These expressions are all equivalent. To specify multiple keywords which require values, enclose the list in parentheses. For example:

```
/BLOCKS=(START:0,END:10)
/PROTECTION=(GR:RW,OWN:RWD)
```

6.3.3 Rules for Entering Output File Qualifiers

Some qualifiers request output from a command and optionally accept a file specification value. For example, the /LIST and /OBJECT qualifiers for the compilers and the assembler, as well as the /EXECUTABLE qualifier for the linker, are output file qualifiers that fit into this category.

The default file specification for output files requested by these qualifiers depends on the placement of the qualifier in the command. The rules are:

- If the qualifier is present by default, the output file specification defaults to the current default device and directory, and the file name of the first, or only, input file. The qualifier provides a default file type. Some examples are:

Command	Output File
LINK A	A.EXE
LINK A,B	A.EXE
LINK [TEST]A,[]B	A.EXE
LINK A.OBJ	A.EXE

- If the qualifier is present following the command name, and the qualifier does not specify an output file specification, the output file specification defaults to the current default device and directory, and the file name of the first, or only, input file. The qualifier provides a default file type. Some examples are:

Command	Output File
LINK/EXE A	A.EXE
LINK/EXE A,B	A.EXE
LINK/EXE A.OBJ	A.EXE

- If the qualifier is present following the specification of an input file, and the qualifier does not specify an output file specification, the output file specification defaults to the device, directory, and file name of the immediately preceding input file. The qualifier provides a default file type. Some examples are:

Command	Output File
FORTRAN A,B/LIST	B.LIS
FORTRAN A+C/LIS,B/LIST+D	C.LIS and B.LIS
FORTRAN [MAL]A/LIST+[]B	[MAL]A.LIS
LINK A+[TEST]D/EXE	[TEST]D.EXE

- If the qualifier specifies a file specification for the output file, then any fields entered in the file specification are used to name the output file, and no default file name is supplied. Some examples are:

Command	Output File
LINK A+B/EXE=C	C.EXE
FORTRAN/LIST=A B+C	A.LIS
FORTRAN/LIST=A B,C	A.LIS (2 versions)
LINK/EXE=[TEST] A	[TEST].EXE

In all cases, the version number of the output file is always one greater than any existing file with the same file name and file type.

Note that when a logical name is used for any input file specifications, the entire equivalence name of the logical name is used to determine the output file specification. If a logical name is used for an input file and its equivalence name contains a file type, then the same file type is used for the output file.

6.4 RULES FOR ENTERING CHARACTER STRING DATA

When you enter commands, you can use any combination of uppercase and lowercase letters. The command interpreter translates lowercase letters to uppercase letters and compresses multiple blank spaces or tabs to a single blank, except when a character string is enclosed in quotation marks (for example, "This is a string.").

Enclose character string data in quotation marks when the string contains any of the following and you do not want the command interpreter to translate them:

- Literal lowercase letters
- Required multiple blanks or tab characters
- Any nonalphanumeric character that has special significance to the command interpreter

The alphanumeric characters are:

A-Z, a-z, 0-9, $, _

Table 6-1 lists the nonalphanumeric characters the command interpreter recognizes and describes their meanings. The characters described as "reserved special characters" can be used in character strings without being enclosed in quotation marks. Other characters may require quotation marks depending on the context of the command. When in doubt, use quotation marks.

Table 6-1
Nonalphanumeric Characters

Symbol	Name	Meaning
@	At sign	Places the contents of a command procedure file in the command input stream
:	Colon	1. Device name delimiter in a file specification; a double colon (::) is a node name delimiter 2. Qualifier value delimiter; separates a qualifier name from a keyword, value, file specification, or character string 3. Symbol name delimiter in a character string assignment 4. Label delimiter
/	Slash	1. Qualifier delimiter 2. Division operator in an arithmetic expression
+	Plus sign	1. Parameter concatenation operator 2. A unary plus sign or addition operator in an arithmetic expression
,	Comma	List delimiter for parameters or for qualifier value lists
-	Hyphen	1. Continuation character 2. A unary minus sign or subtraction operator in an arithmetic expression
( )	Parentheses	1. List delimiters for qualifier value list 2. Operation precedence indicators in an arithmetic expression 3. Lexical function argument delimiters
[ ]	Square brackets	1. Directory name delimiters in a file specification 2. Substring specification delimiter in an assignment statement
< >	Angle brackets	Directory name delimiters in a file specification
?	Question mark	Reserved special character

(continued on next page)

Table 6-1 (Cont.)
Nonalphanumeric Characters

Symbol	Name	Meaning
&	Ampersand	Execution-time substitution operator; otherwise, a reserved special character
\	Back slash	Reserved special character
=	Equal sign	1. Qualifier value delimiter; separates a qualifier name from a keyword, value, or string option 2. Symbol name delimiter for an assignment statement
^	Circumflex	Reserved special character
#	Number sign	Reserved special character
*	Asterisk	1. Wild card in a file specification 2. Multiplication operator in an arithmetic expression
'	Apostrophe	Substitution operator
.	Period	1. File type and version number delimiters in a file specification 2. Operation delimiter in an arithmetic expression
;	Semicolon	Version number delimiter in a file specification
%	Percent sign	Radix operator; otherwise, a reserved special character
!	Exclamation point	Comment delimiter; the command interpreter ignores the remainder of the command line
"	Quotation mark	String delimiter

6.5 RULES FOR ENTERING NUMERIC VALUES

The command interpreter treats all literal numeric values as decimal integers. To specify numeric values in commands, you can use one of the following radix operators preceding a numeric value to indicate the radix (number base):

Operator	Meaning
%D	Decimal
%X	Hexadecimal
%O	Octal

For example:

```
%D16
%X10
%O20
```

These values are equivalent.

There cannot be any blanks between a radix operator and a value.

6.6 RULES FOR FORMING EXPRESSIONS

When the command interpreter evaluates expressions, it assigns a value based on the result of the operations specified in the expression. If the expression contains logical operators or arithmetic or string comparison operators, the expression is considered true if it results in an odd numeric value; the expression is considered false if it results in an even numeric value.

The following sections show how to specify expressions using assignment statements. Symbols defined by assignment statements can be tested in IF commands. The rules defined below also apply to the specification of expressions in the IF command.

6.6.1 Rules for Entering Operators

Logical and comparison operators must be preceded by a period (.) with no intervening blanks. The operator must be terminated with a period. You can type any number of blanks or tabs between operators and operands. For example, the following expressions are equivalent:

```
A.EQS.B
A .EQS.  B
```

Each operator (with the exception of .NOT.) must have operands on each side.

6.6.2 Rules for Specifying Logical Operations

Use logical operators to perform logical functions on arithmetic values or to construct complicated expressions. Some examples are:

Expression	Result
A = 3 .OR. 5	A = 7
B = 3 .AND. 5	B = 1
C = .NOT.3	C = -4
D = 3 + 4 .OR. 2 + 4	D = 7

Operands for logical operations must be literal numeric values, symbol names equated to numeric values, or expressions.

6.6.3 Rules for Specifying Arithmetic Comparisons

Use arithmetic comparison operators to compare numeric values. If the result of an arithmetic comparison is true, the expression has a value of 1; if the result of the comparison is false, the expression has a value of 0. Some examples are:

Expression	Result
A = 1.LE.2	A = 1
B = 1.GT.2	B = 0
C = 1 + 3 .EQ. 2 + 5	C = 0
D = "TRUE".EQ.1	D = 1
E = "FALSE"	E = 0

Operands in arithmetic comparisons can be literal numeric values; symbol names equated to numeric values, expressions that yield numeric values; or character strings enclosed in quotation marks that begin with the uppercase letters T, Y, F, or N. (A character string beginning with the letters T or Y has a value of 1; a character string beginning with the letters F or N has a value of 0.)

6.6.4 Rules for Specifying String Comparisons

Use string comparison operators to compare alphanumeric character strings. Character string comparison is based on the binary value of the characters in the string; comparison is on a character-by-character basis. If one string is longer than the other, the shorter string is padded on the right with blanks before the comparison is made. Lowercase letters have higher numeric values than uppercase letters. If the result of a comparison is true, the symbol name is given a value of 1; if the comparison is false, the symbol name is given a value of 0. Some examples are:

Expression	Result
A = "MAYBE".LTS."maybe"	A = 1
B = "ABCD".LTS."EFG"	B = 1
C = "YES".GTS."YESS"	C = 0
D = "AAB".GTS."AAA"	D = 1

Operands in string comparisons can be literal strings enclosed in quotation marks, symbol names equated to character strings, or literal numeric values. If you do not enclose a literal character string in quotation marks, the command interpreter assumes the string is a symbol name and issues an error message if the symbol is not defined.

6.6.5 Rules for Specifying Arithmetic Operations

Use arithmetic operators to perform calculations on numeric integer values. Some examples are:

Expression	Result
A = 5 + 10 / 2	A = 10
B = 5 * 3 - 4 * 6 / 2	B = 3
C = 5 * (6 - 4) - 8 / (2 - 1)	C = 2
D = %X50	D = 80
E = %X10 + 5	E = 21

Note that nondecimal values (specified by radix operators) are converted to decimal.

Operands in arithmetic comparisons can be literal numeric values, symbol names equated to numeric values, or expressions that yield numeric values.

6.6.6 Rules of Precedence in Expressions

When you specify more than one operation in an expression, the operations are performed in the order of precedence listed below, where 1 is the lowest precedence and 6 is the highest precedence. For example, multiplication (precedence 6) is performed before addition (precedence 5).

Operations of the same precedence are performed from left to right, as they appear in the command.

Operator	Precedence	Operation
.OR.	1	Logical OR
.AND.	2	Logical AND
.NOT.	3	Logical complement
.EQ.	4	Arithmetic equal to
.GE.	4	Arithmetic greater than or equal to
.GT.	4	Arithmetic greater than
.LE.	4	Arithmetic less than or equal to
.LT.	4	Arithmetic less than
.NE.	4	Arithmetic not equal to
.EQS.	4	String equal to
.GES.	4	String greater than or equal to
.GTS.	4	String greater than
.LES.	4	String less than or equal to
.LTS.	4	String less than
.NES.	4	String not equal to
+	5	Arithmetic sum
-	5	Arithmetic difference
*	6	Arithmetic product
/	6	Arithmetic quotient

Use parentheses for grouping to override the order in which operators are evaluated; expressions within parentheses are evaluated first.

6.7 RULES FOR SPECIFYING LEXICAL FUNCTIONS

You can use lexical functions in any context in which you normally use symbols or expressions. The general format of a lexical function is:

```
'F$function-name([args,...])
```

'F$
indicates that what follows is a lexical function. The substitution operator (') is required.

function-name
specifies the function to be evaluated. All function names are keywords. You can truncate function names to any unique truncation.

( )
enclose function arguments, if any. The parentheses are required for all functions, including functions that do not accept any arguments.

[args,...]
specify arguments for the function. You can specify arguments using symbol names, numeric literals, or string literals enclosed in quotation marks. The command interpreter assumes that all strings beginning with alphabetic letters that are not enclosed in quotation marks are symbol names. If a symbol is undefined, the command interpreter substitutes a null string.

Function arguments cannot specify string substitution or other lexical functions.

For a complete list of the lexical functions, their formats, and the arguments required by each, see Table 5-1.

6.8 RULES FOR ENTERING DATES AND TIMES

When a command accepts a qualifier that specifies a time value, the time value is either an absolute time or a delta time:

- An absolute time is a specific date and time of day, for example 10-JUN-1978 10:53:22.10.
- A delta time is a future offset from the current date and time of day, for example 2 days and 3 hours from now.

The syntax rules for specifying time values are described below.

6.8.1 Absolute Times

Absolute times have the format:

[dd-mmm-yyy[:]][hh:mm:ss.ss]

You can specify either the date or the time, or both. The variable fields are as follows:

Field	Meaning
dd	Day of month (1 through 31)
mmm	Month; the month must be specified as one of the following three-character abbreviations: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC
yyyy	Year
hh	Hour of the day (0 through 23)
mm	Minute of the hour (0 through 59)
ss.ss	Seconds and hundredths of seconds (00.00 through 59.99)

6.8.1.1 **Syntax** - The punctuation marks indicate how the system interprets the time value you enter, as follows:

1. If you specify both the date (dd-mmm-yyyy) and the time (hh:mm:ss.ss), you must type the colon between the date and the time.

2. You can truncate either the date or the time on the right; however, if you are specifying both a date and a time, the date part must contain at least one hyphen.

3. You can omit any of the fields within the date or time, as long as you type the punctuation marks; the system supplies default values.

6.8.1.2 **Defaults** - When the date or any of its fields is omitted from an absolute time value, the system supplies the current day, month, and year by default.

When any field is omitted from the time, the system supplies a value of 0 for the field.

6.8.1.3 **Examples** - Some examples of specifying absolute times are:

Time Specification	Result
28-JUN-1978:12	12:00 noon on June 28, 1978
28-JUN	Midnight (00:00 o'clock) at the beginning of the 28th of June, this year
15	3:00 p.m., today
15-	The 15th day of the current month and year, at midnight
18:30	6:30 p.m., today
15--::30	00:30 o'clock, on the 15th day of the current month

Note that whenever you issue a command and specify an absolute time that has already passed, the system executes the specified action immediately.

6.8.2 **Delta Times**

Delta times have the format:

[dddd-][hh:mm:ss.ss]

The variable fields are as follows:

Field	Meaning
dddd	Number of days, 24-hour units (0 through 9999)
hh	Number of hours (0 through 23)
mm	Number of minutes (0 through 59)
ss.ss	Number of seconds and hundredths of seconds (00.0 through 59.99)

6.8.2.1 **Syntax** - When you specify a delta time value, you can truncate the time field on the right. You can also omit any of the variable fields, as long as you supply the punctuation marks.

6.8.2.2 **Defaults** - When any field is omitted from a delta time value, the system supplies a value of 0 for the field.

6.8.2.3 **Examples** - Some examples of specifying delta times are:

Time Specification	Result
3-	3 days from now (72 hours)
3	3 hours from now
:30	30 minutes from now
3-:30	3 days and 30 minutes from now
15:30	15 hours and 30 minutes from now

PART II

COMMAND DESCRIPTIONS

= (Assignment Statement)

Defines a symbolic name for a character string or for an arithmetic value or expression.

Formats

```
symbol-name =[=] expression

symbol-name :=[=] string

symbol-name[bit-position,size]=[=] expression[1]

symbol-name[offset,size]:=[=]  string[1]
```

Command Qualifiers

None.

Prompts

None.

Command Parameters

symbol-name

Defines a 1- to 255-alphanumeric character string name for the symbol. The symbol name must begin with an alphabetic character (uppercase and lowercase characters are equivalent).

If you specify a single equal sign (=) in the assignment statement, the symbol name is placed in the local symbol table for the current command level.

If you specify double equal signs (==) in the assignment statement, the symbol name is placed in the global symbol table. Global symbols are recognized in any command procedure as well as at the interactive command level. There cannot be any blanks between the equals signs or between an equals sign and a colon.

expression

Specifies a numeric value or arithmetic expression to be equated to the symbol name. If you specify an expression, the command interpreter analyzes the expression, substituting symbol values if necessary, before making the assignment.

You can form arithmetic expressions using any of the logical or arithmetic operators listed below. Combinations of operations are evaluated in the order of precedence indicated, where 1 is the lowest precedence and 6 is the highest. Operators of the same precedence are evaluated left to right. Use parentheses for grouping to override the order in which operators are evaluated; expressions within parentheses are evaluated first.

---

[1] In this syntax, the brackets in the expressions [bit-position,size] and [offset,size] are required. The brackets in the expressions =[=] and :=[=] indicate that the second equal sign is optional.

Operator	Precedence	Operation
.OR.	1	Logical OR
.AND.	2	Logical AND
.NOT.	3	Logical complement
.EQ.	4	Arithmetic equal to
.GE.	4	Arithmetic greater than or equal to
.GT.	4	Arithmetic greater than
.LE.	4	Arithmetic less than or equal to
.LT.	4	Arithmetic less than
.NE.	4	Arithmetic not equal to
.EQS.	4	String equal to
.GES.	4	String greater than or equal to
.GTS.	4	String greater than
.LES.	4	String less than or equal to
.LTS.	4	String less than
.NES.	4	String not equal to
+	5	Arithmetic sum
-	5	Arithmetic difference
*	6	Arithmetic product
/	6	Arithmetic quotient

For further details on the syntax requirements and valid ways of specifying expressions, see Section 6.6, "Rules for Forming Expressions."

string

Specifies a character string value (or any expression resulting in a character string value) to be equated to the symbol. The string can contain any alphanumeric or special characters; enclose it in quotation marks if it contains any multiple blanks or tab characters, lowercase letters, an exclamation point (!), or quotation marks. To specify a string that contains literal quotation marks, enclose the entire string in quotation marks, and use a double set of quotation marks within the string. For example:

```
$ HELLO := "PATTI SAYS ""HI"""
```

You can specify a null string either by using a double set of quotation marks with no intervening characters or by specifying no string. For example:

```
$ NULLSTRING := ""
```

You can omit a trailing quotation mark on the end of a line.

If a string beginning with a dollar sign is not enclosed in quotation marks, the command interpreter assumes that the string is the file specification of an executable image. When the symbol-name thus defined appears as the first item in a command string, the command interpreter executes the image. The file specification must include a device name.

[bit-position,size]

Defines a range within the current value of symbol-name that is to be stored with the binary value of the indicated expression.

[offset,size]

Defines the substring location at which the current value of symbol-name is to be overlayed with characters from the indicated string expression.

offset
: Specifies the position relative to the beginning of the symbol-name's string value at which replacement is to begin. If the offset is greater than the length of the string, the resulting string is filled with blanks to the requested offset before replacement occurs.

size
: Specifies the number of characters, beginning with the first character, in the string expression to extract. If the size is greater than the length of the string, the string is blank-filled on the right to the requested size before it is used to overlay the symbol-name string.

You can specify the offset and size using literal numeric values or arithmetic expressions, including expressions consisting of lexical functions.

When you specify a substring expression in an assignment statement, there cannot be any blanks preceding or following the substring expression.

Description

Symbols defined via assignment statements allow you to extend the command language. At the interactive command level, you can define synonyms for commands or command lines. In command procedure files, you can define symbols and test their values to provide for conditional execution and substitution of variables.

You can also define symbols in the following ways:

- By passing parameters to a command procedure when it is executed by the @ (Execute Procedure) command or the SUBMIT command. These symbols, named P1, P2, and so on through P8, are always local to the command procedure to which they are passed.

- By using the INQUIRE command within a command procedure to prompt for the assignment of a value during the execution of the procedure. A symbol defined with the INQUIRE command can be either local or global.

The SHOW SYMBOL command displays the current assignment values of symbols defined as local or global symbols.

Reserved Symbols: The following symbol names are reserved. They are global symbols under the control of the operating system and cannot be explicitly redefined:

$STATUS
: Return status from the most recently executed command or program, or status value specified by a command procedure when it exited.

$SEVERITY
: Severity level of the status code from the most recently executed command or program. The possible values are:

Value	Severity Level
0	Warning
1	Success
2	Error
3	Informational
4	Severe, or fatal error

You can use these symbolic names to test the completion status of command or program execution to determine the success or failure of a request, and optionally to provide for conditional processing based on the value returned.

Symbol Substitution: When the command interpreter performs symbol substitution, it searches the local symbol table for the current command level for a match and replaces the symbol name with the numeric value or character string assigned to it. If the symbol is not found, the command interpreter searches the local symbol tables of preceding command levels, in reverse order. Last, it searches the global symbol table.

The command interpreter substitutes symbols when:

- A symbol name appears as the first word in a command and the next nonblank character is not an equal sign (=) or a colon (:).
- A symbol name in a command string is enclosed in apostrophes ('). If the symbol name appears in a string enclosed in quotation marks, precede the symbol name with two apostrophes and terminate it with one apostrophe. If a symbol is undefined, it is treated as a null string.
- It executes a command that automatically evaluates expressions in particular contexts, for example, in the IF, DEPOSIT, EXAMINE, or WRITE commands; on the right-hand side of arithmetic assignment statements; and in lexical functions. If a symbol is undefined, an error occurs.
- A symbol name in a command string is preceded with an ampersand character (&). If a symbol is undefined, it is treated as a null string within the context of the command that is being executed.

The command interpreter substitutes symbols during two distinct phases of command processing:

1. Symbol names at the beginning of a command string and symbol names that are delimited with apostrophes (') are substituted while the command interpreter scans the command, but before the command is executed.
2. Symbol names occurring within contexts where substitution is automatic and symbol names that are preceded with ampersands (&) are substituted during the execution of the command.

Each time substitution occurs, substitutions are made from left to right on each line in the command.

Examples

1.
```
$ TIME := SHOW TIME
$ TIME
  28-JUL-1978 11:55:44
```

The symbol TIME is equated to the command string SHOW TIME. Because the symbol name appears as the first word in a command string, the command interpreter automatically substitutes it with its string value and executes the command SHOW TIME.

2.
```
$ LIST :== DIRECTORY
$ TIME :== SHOW TIME
$ QP :== SHOW QUEUE/DEVICE
$ SS :== SHOW SYMBOL
```

The file, SYNONYM.COM, contains the assignment statements shown; these are user-defined synonyms for commands. Execute this command procedure as follows:

```
@SYNONYM
```

The global symbol definitions are made, and you can now use these synonyms at the interactive command level. Note that the assignments are global; otherwise, the symbol names would be deleted after the file SYNONYM.COM completed execution.

3.
```
$ COUNT = 0
$ LOOP:  COUNT = COUNT + 1
    .
    .
$ IF COUNT.LT.5 THEN GOTO LOOP
```

The symbol COUNT is initially assigned a numeric value of 0; a loop is established to increment the value of COUNT by 1 each time the loop is entered. Note that when the symbol name COUNT appears on the right-hand side of an arithmetic assignment statement, the command interpreter automatically substitutes its current value.

The IF command tests the value of COUNT; if less than 5, the procedure branches to the label LOOP and the statements between the label LOOP through the IF command are executed again. When the value of the symbol count reaches 5, the loop is not executed again and the command following the IF command is executed.

4.
```
$ NAME := MYFILE
$ TYPE := .DAT
$ PRINT 'NAME''TYPE'
```

In this example the symbol NAME is equated to a file name and the symbol TYPE is equated to a file type. The PRINT command refers to both of these symbol names to form a file specification. The apostrophes surrounding each symbol name indicate that these are symbols to be substituted.

The PRINT command prints the file MYFILE.DAT.

5.
```
$ STAT := $DBA1:[CRAMER]STAT
$ STAT
```

When a symbol is equated to a string that begins with a dollar sign followed by a file specification, the command interpreter assumes that the file specification is that of an executable image, that is, the file has a file type of EXE. The symbol STAT in this example becomes a synonym for the command:

```
$ RUN DBA1:[CRAMER]STAT.EXE
```

When you type STAT, the command interpreter executes the image.

6.
```
$ WRITE SYS$OUTPUT "Beginning Test No.  ''COUNT'"
```

The WRITE command writes a line of data to the current output device. The string to be written is enclosed in quotation marks so that the lowercase letters will not be translated to uppercase. However, the string contains the name of a symbol, COUNT, which must be substituted with its current value before the line is written.

The symbol COUNT is preceded with two apostrophes and terminated with a single apostrophe to request that symbol substitution be performed.

7.
```
$ COUNT = 0
$ LOOP:
$ COUNT = COUNT + 1
$ IF P'COUNT' .EQS.  "" THEN EXIT
$ APPEND/NEW &P'COUNT' SAVE.ALL
$ DELETE &P'COUNT';*
$ IF COUNT .NE.  8 THEN GOTO LOOP
$ EXIT
```

This command procedure uses a counter to refer to parameters that are passed to it. Up to eight parameters, named P1, P2, and so on, can be passed. Each time through the loop, the procedure uses an IF command to check whether the value of the current parameter is a null string. When the IF command is scanned, the symbol COUNT is substituted with its current value and concatenated with the letter P. The first time through the loop, the IF command tests P1; the second time through the loop it tests P2, and so on. The substitution of P1, P2, and so on, is automatic within the context of the IF command, because the IF command tests symbolic and literal expressions.

The APPEND and DELETE commands, however, do not automatically perform any substitution, because they expect and require file specifications as input parameters. The ampersand (&) precedes the P'COUNT' expression for these commands. When these commands are initially scanned each time through the loop, COUNT is substituted with its current value. Then, when the commands execute, the & causes another substitution: P1, P2, and so on. If this procedure is invoked with the line:

```
$ @COPYDEL ALPHA.TXT BETA.DOC
```

The files ALPHA.TXT and BETA.DOC are each appended to the file SAVE.ALL and then deleted.

8.
```
$ FILE_SPEC := 'P1'
$ LOC = 'F$LOCATE(".",FILE_SPEC)
$ FILE_NAME := 'F$EXTRACT(0,LOC,FILE_SPEC)
```

These lines show how to extract the file name portion of a string containing a file name, file type, and optionally a file version number.

The first statement equates the symbol FILE_SPEC to the parameter P1, which must be passed to the command procedure. Note that the apostrophes are required: otherwise, the symbol name FILE_SPEC is equated to the literal string P1.

The second statement uses the lexical function F$LOCATE to locate the period within the file specification string. The function returns, in the symbol LOC, a numeric value representing the offset of the period within the string value of FILE_SPEC.

The third statement uses the symbol name LOC to specify how many characters of the file specification are to be extracted. The lexical function F$EXTRACT requests that LOC characters, beginning at the beginning of the string, be extracted from the current value of the string FILE_SPEC. The result is equated to the symbol name FILE_NAME.

If this procedure is passed the parameter MYFILE.DAT, the resulting value of the symbol LOC is 6 and the resulting value of the symbol FILE_NAME is the string MYFILE.

9.
```
$ FILE_NAME[0,2]:= OL
```

The substring expression in the assignment statement overlays the first two characters of a file name string with the letters OL. The offset of 0 requests that the overlay begin with the first character in the string, and the size specification of 2 indicates the number of characters to overlay.

If the current value of the symbol FILE_NAME is MYFILE when this assignment statement is executed, the resulting value of the symbol name is OLFILE.

10.
```
$ FILE_TYPE := .TST
$ FILE_NAME['F$LENGTH(FILE_NAME),4)]:= 'FILE_TYPE'
```

In this example, the symbol name FILE_TYPE is equated to the string .TST. The second assignment statement uses the lexical function F$LENGTH to define the offset value in the substring expression.

The F$LENGTH lexical function returns the length of the string equated to the symbol FILE_NAME; thus the substring expression requests that 4 characters of the string currently equated to the symbol FILE_TYPE be placed at the end of the string currently equated to FILE_NAME. If the current value of FILE_NAME is MYFILE, the F$LENGTH logical function returns a value of 6 and the substring expression is:

```
$ FILE_NAME[6,4]:= 'FILE_TYPE'
```

Thus, the resultant value of the string FILE_NAME is MYFILE.TST.

@ (Execute Procedure)

Executes a command procedure or requests the command interpreter to read subsequent command input from a specific file or device.

Format

```
@file-spec  [p1 [p2 [... p8]]]

   Command Qualifiers

   /OUTPUT=file-spec
```

Prompts

None.

Command Parameters

file-spec
Specifies the command procedure to be executed, or the device or file from which input for the preceding command is to be read.

If you do not specify a file type, the system uses the default file type of COM.

p1 p2 ... p8
Specify from one to eight optional parameters to pass to the command procedure. The parameters assign numeric or character string values to the symbols named P1, P2, and so on in the order of entry, to a maximum of 8. The symbols are local to the specified command procedure. The command interpreter sets all unspecified parameters to null strings.

Separate each parameter with one or more blanks. You can specify a numeric value for a parameter using any valid arithmetic expression. You can also specify a character string value using any alphanumeric or special characters, with the following restrictions:

1. If the first parameter begins with a slash character (/), you must enclose the parameter in quotation marks.

2. To pass a parameter that contains embedded blanks or literal lowercase letters, place the parameter in quotation marks.

3. To pass a parameter that contains literal quotation marks, enclose the entire string in quotation marks and use a double set of quotation marks within the string. For example:

```
$ @TEST "Never say ""quit"""
```

When the procedure TEST.COM executes, the parameter P1 is equated to the string:

```
Never say "quit"
```

Description

Use command procedures to catalog frequently used sequences of commands. A command procedure can contain:

- Any valid DCL command. All commands must begin with a dollar sign ($) character. If a command is continued with the continuation character (-), the subsequent lines must not begin with a $.

- Data. Any line in a command procedure that does not contain a dollar sign in the first character position (or is not a continuation line) is treated as input data for the command or program that is currently executing. The DECK command allows you to specify that data contains dollar signs in record position one.

- Qualifiers and/or parameters for a specific command. If the file contains only parameters for the command, the @ command must be preceded by a space. If the file contains qualifiers for the command, the @ command must not be preceded with a space. If the file contains only parameters and/or qualifiers, the lines must not begin with dollar signs ($). Any additional data on the command line following the @file-spec is treated as parameters for the procedure.

A command procedure can also contain a request to execute another command procedure. The maximum level to which command procedures can be nested is eight.

Command procedures can also be queued for processing as batch jobs, either with the SUBMIT command or by placing a deck of cards containing the command procedure in the system card reader. Batch jobs submitted through the card reader must be preceded with JOB and PASSWORD commands.

For more information and examples of creating and submitting command procedures for execution, see Chapter 5, "Command Procedures and Batch Jobs."

Command Qualifiers

/OUTPUT=file-spec

Requests that all output directed to the logical device SYS$OUTPUT be written to the file or device specified. System responses and error messages are written to SYS$COMMAND as well as to the specified file.

If you specify /OUTPUT, the qualifier must immediately follow the file specification of the command procedure. Otherwise, it is interpreted as a parameter to pass to the command procedure.

Examples

1.
```
$ ON WARNING THEN EXIT
$ IF P1.EQS."" THEN INQUIRE P1 FILE
$ FORTRAN/LIST 'P1'
$ LINK 'P1'
$ RUN 'P1'
$ PRINT 'P1'
```

This command procedure, named DOFOR.COM, executes the FORTRAN, LINK, and RUN commands to compile, link, and execute a program whose file specification is passed as a parameter. The ON command requests that the procedure not continue if any of the commands results in warnings or errors. The IF command checks to see if a parameter was passed; if not, the INQUIRE command issues a prompting message to the terminal and equates what you enter with the parameter P1. You can execute this procedure as follows to compile, link, run, and obtain a listing of the program AVERAGE.FOR:

```
$ @DOFOR AVERAGE
```

2.
```
$ @MASTER/OUTPUT=MASTER.LOG
```

This command executes a procedure named MASTER.COM; all output is written to the file MASTER.LOG.

3.
```
$ RUN 'P1' -
   /BUFFER_LIMIT=1024 -
   /FILE_LIMIT=4 -
   /PAGE_FILES=256 -
   /QUEUE_LIMIT=2 -
   /SUBPROCESS_LIMIT=2 -
   'P2'  'P3'  'P4'  'P5'  'P6'  'P7'  'P8'
```

This procedure, named SUBPROCES.COM, issues the RUN command to create a subprocess. It contains qualifiers defining quotas for subprocess creation. For example, the procedure can be invoked as follows:

```
$ @SUBPROCES  LIBRA  /PROCESS_NAME=LIBRA
```

In this example, LIBRA is equated to P1; it is the name of an image to execute in the subprocess. /PROCESS_NAME=LIBRA is equated to P2; it is a qualifier for the RUN command.

4.
```
$ ASSIGN SYS$COMMAND:  SYS$INPUT:
$ NEXT:
$ INQUIRE NAME "File name"
$ IF NAME.EQS."" THEN EXIT
$ EDIT 'NAME'.DOC
$ GOTO NEXT
```

This procedure, named EDOC.COM, invokes the default system editor, SOS. When an edit session is terminated, the procedure loops to the label NEXT. Each time through the loop, the procedure requests another file name for the editor and supplies the default file type of DOC. When a null line is entered in response to the INQUIRE command, the procedure terminates with the EXIT command.

The ASSIGN command changes the equivalence of SYS$INPUT for the procedure: this allows the editor to read input from the current command device (the terminal) rather than from the input stream (the command procedure file).

ALLOCATE

Provides exclusive access to a device and optionally establishes a logical name for the device. Once a device has been allocated, other users cannot access the device until you specifically deallocate it or log out.

Format

```
ALLOCATE  device-name[:]  [logical-name[:]]

    Command Qualifiers

    None.
```

Prompts

Device: device-name

Log_name: logical-name

Command Parameters

device-name
: Specifies the name of the device to be allocated. The device name can be a generic device name, such that if no controller or unit number is specified, the system allocates any available device that satisfies those components of the device name that are specified.

logical-name
: Specifies a 1- to 63-character logical name to be associated with the device. The logical name is placed in the process logical name table, with the name of the physical device allocated as its equivalence name. Subsequent references to the logical name result in automatic translation to the specified device name.

  If you specify a trailing colon (:) on the logical name, the colon is removed from the name before the name is placed in the logical name table.

Examples

1.
```
$ ALLOCATE _DMB2:
_DMB2: ALLOCATED
```

The ALLOCATE command requests the allocation of a specific RK06 disk drive, that is, unit 2 on controller B. The response from the ALLOCATE command indicates that the device was successfully allocated.

2.
```
$ ALLOCATE MT:   TAPE:
_MTB2: ALLOCATED
  .
  .
$ SHOW LOGICAL TAPE
  TAPE = _MTB2:     (Process)
$ DEALLOCATE TAPE
$ DEASSIGN TAPE
```

The ALLOCATE command requests the allocation of any tape device, to be assigned the logical name, TAPE. The ALLOCATE command locates an available tape device and responds with the name of the device allocated. Subsequent references to the device TAPE in user programs or command strings are translated to the device name MTB2.

When the tape device is no longer needed, the DEALLOCATE command deallocates it and the DEASSIGN command deletes the logical name. Note that the logical name, TAPE, was specified with a colon on the ALLOCATE command, but that the logical name table entry does not have a colon.

APPEND

Adds the contents of one or more specified input files to the end of a specified output file.

Format

```
APPEND  input-file-spec,...  output-file-spec

   Command Qualifiers          Default

   /[NO]LOG                    /NOLOG

   File Qualifiers

   /ALLOCATION=n
   /[NO]CONTIGUOUS             /NOCONTIGUOUS
   /EXTENSION=n
   /FILE_MAXIMUM=n
   /[NO]NEW                    /NONEW
   /PROTECTION=code
   /[NO]READ_CHECK             /NOREAD_CHECK
   /[NO]WRITE_CHECK            /NOWRITE_CHECK
```

Prompts

From: input-file-spec,...

To: output-file-spec

Command Parameters

input-file-spec,...
Specifies the names of one or more input files to be appended.

If you specify more than one input file, separate the specifications with either commas (,) or plus signs (+). Commas and plus signs are equivalent: all files specified are appended, in the order specified, to the end of the output file.

You can use a wild card (*) in place of the file name, type, or version field. Then, all files that satisfy the remaining components are appended.

output-file-spec
Specifies the name of the file to which the input files are to be appended.

You must specify at least one field in the output file specification. If you do not specify a device and/or directory, the APPEND command uses your current default device and directory. For other fields that you do not specify, the APPEND command uses the corresponding field of the input file specification.

If you specify a wild card in any field(s) of the output file specification, the APPEND command uses the corresponding field of the related input file specification.

Description

The APPEND command is similar in syntax and function to the COPY command. Normally, the APPEND command adds the contents of one or more files to the end of an existing file without incrementing the version number. You can use the /NEW qualifier to request that if the output file does not exist, the APPEND command should create it.

Command Qualifiers

/LOG
/NOLOG

Controls whether the APPEND command displays the file specifications of each file appended.

If you specify /LOG, the APPEND command displays, for each append operation, the file specifications of the input and output files, and the number of blocks or the number of records appended. At the end of command processing, the APPEND command displays the number of new files created.

File Qualifiers

/ALLOCATION=n

Forces the initial allocation of the output file to the number of 512-byte blocks specified as n.

This qualifier is valid only if /NEW is specified, and the allocation size is applied only if a new file is actually created. If a new file is created and you do not specify /ALLOCATION, the initial allocation of the output file is determined by the size of the input file.

/CONTIGUOUS
/NOCONTIGUOUS

Indicates whether the output file is contiguous, that is, whether the file must occupy consecutive physical disk blocks.

By default, the APPEND command creates an output file in the same format as the corresponding input file. If an input file is contiguous, the APPEND command attempts to create a contiguous output file, but does not report an error if there is not enough space. If you append multiple input files of different formats, the output file may or may not be contiguous. Use the /CONTIGUOUS qualifier to ensure that files are contiguous.

/EXTENSION=n

Specifies the number of blocks to be added to the output file each time the file is extended.

The extension value is applied only if a new file is actually created. If you specify /EXTENSION, the /NEW qualifier is assumed.

/FILE_MAXIMUM=n

Specifies the maximum number of logical records that the output file can contain.

This qualifier is valid only for new relative files. If you specify /FILE_MAXIMUM, the /NEW qualifier is assumed.

/NEW
/NONEW

Controls whether the APPEND command creates a new file. By default, the output file specified must already exist. Use /NEW to request that if the specified output file does not already exist, the APPEND command should create it.

/PROTECTION=code

Defines the protection to be applied to the output file.

Specify the protection code using the standard rules for specifying protection (these rules are given in the discussion of the SET PROTECTION command). Any protection attributes not specified are taken from the current protection of the output file; or, if a new file is created, from the current default protection.

/READ_CHECK
/NOREAD_CHECK

Requests the APPEND command to read each record in the input file(s) twice to verify that all records were correctly read.

/WRITE_CHECK
/NOWRITE_CHECK

Requests the APPEND command to read each record in the output file after it is written to verify that the record was successfully appended and that the file can subsequently be read without error.

Examples

1. `$ APPEND  TEST.DAT  NEWTEST.DAT`

   The APPEND command appends the contents of the file TEST.DAT from the default disk and directory into the file NEWTEST.DAT.

2.

```
$ APPEND/NEW/LOG  *.TXT  MEMO.SUM
%APPEND-I-CREATED, DBA2:[MALCOLM]MEMO.SUM;1 created
%APPEND-S-COPIED, DBA2:[MALCOLM]ALPHA.TXT;2 copied to DBA2:[MALCOLM]MEMO.SUM;1 (1 block)
%APPEND-S-APPENDED, DBA2:[MALCOLM]BETA.TXT;3 appended to DBA2:[MALCOLM]MEMO.SUM;1 (3 records)
%APPEND-S-APPENDED, DBA2:[MALCOLM]GAMMA.TXT;7 appended to DBA2:[MALCOLM]MEMO.SUM;1 (51 records)
%APPEND-S-NEWFILES, 1 file created
```

   The APPEND command appends all files with file types of TXT to a file named MEMO.SUM. The /LOG qualifier requests a display of the specifications of each input file appended. If the file MEMO.SUM does not exist, the APPEND command creates it, as the output shows.

3.

```
$ APPEND/LOG  A.DAT, B.MEM  C.*
%APPEND-S-APPENDED, DBA2:[MALCOLM]A.DAT;4 appended to DBA2:[MALCOLM]C.DAT;4 (2 records)
%APPEND-S-APPENDED, DBA2:[MALCOLM]B.MEM;5 appended to DBA2:[MALCOLM]C.DAT;4 (29 records)
```

   The input file specifications in this example request APPEND to append separate files. The APPEND command appends the files A.DAT and B.MEM to the file C.DAT.

4.

```
$ APPEND/LOG   A.* B.*
%APPEND-S-APPENDED, DBA2:[ANKLAMJA.DAT;5 appended to DBA2:[ANKLAMJB.DAT;1 (5 records)
%APPEND-S-APPENDED, DBA2:[ANKLAMJA.DOC;2 appended to DBA2:[ANKLAMJB.DAT;1 (1 record)
```

Both the input and output file specifications contain wild cards in the file type field. The APPEND command appends each file with a file name of A to a file with a file name of B; the file type of the first input file located determines the output file type.

ASSIGN

Equates a logical name to a physical device name; to a complete file specification, or to another logical name; and places the equivalence name string in the process, group, or system logical name table.

Format

```
ASSIGN   device-name[:]  logical-name[:]

   Command Qualifiers        Default

   /GROUP                    /PROCESS
   /PROCESS                  /PROCESS
   /SUPERVISOR_MODE          /SUPERVISOR_MODE
   /SYSTEM                   /PROCESS
   /USER_MODE                /SUPERVISOR_MODE
```

Prompts

Device: device-name

Log_name: logical-name

Command Parameters

device-name

Specifies the name of the device or file specification to be assigned a logical name.

If you specify a physical device name, terminate the device name with a colon (:).

You can specify a logical name for the device name. If the logical name translates to a device name, and will be used in place of a device name in a file specification, terminate it with a colon (:).

logical-name

Specifies a 1- to 63-character logical name to be associated with the device. If you terminate the logical name with a colon, the system removes the colon before placing the name in a logical name table. By default, the logical name is placed in the process logical name table.

If the logical name contains any characters other than alphanumeric characters or delimiters not recognized within device names, enclose it in quotation marks.

If the logical name already exists in the specified logical name table, the new definition supersedes the old definition, and the system displays an informational message indicating that fact.

Description

If you enter more than one of the qualifiers /PROCESS, /GROUP, or /SYSTEM, or both of the qualifiers /SUPERVISOR_MODE and /USER_MODE, only the last one entered is accepted.

For additional information on how to create and use logical names, see Section 2.2 "Logical Names."

Command Qualifiers

/GROUP
Places the logical name and its associated device name in the group logical name table. Other users with the same group number in their UICs (user identification codes) can access the logical name.

The user privilege GRPNAM is required to place a name in the group logical name table.

/PROCESS
Places the logical name and its associated device name in the process logical name table. This is the default.

/SUPERVISOR_MODE
Specifies, for an entry in the process logical name table, that the logical name be entered in supervisor mode.

This is the default for the process logical name table entries. The /SUPERVISOR_MODE qualifier is ignored when entries are made in the group or system logical name tables.

/SYSTEM
Places the logical name and its associated device name in the system logical name table. Any user can access the logical name.

The user privilege SYSNAM is required to place a name in the system logical name table.

/USER_MODE
Specifies, for an entry in the process logical name table, that the logical name be entered in the user mode.

A user mode logical name is practical for the execution of a single image; it allows an image executing in a command procedure to redefine SYS$INPUT. User mode entries are deleted when any image executing in the process exits (that is, after any DCL command or user program that executes an image completes execution), or when a STOP command is issued.

By default, process logical name table entries are made in supervisor mode. The /USER_MODE qualifier is ignored when entries are made in the group or system logical name tables.

Examples

1.
```
$ ASSIGN DBA2:[CHARLES]  CHARLIE
$ PRINT  CHARLIE:TEST.DAT
```

The ASSIGN command associates the logical name CHARLIE with the directory name CHARLES on the disk DBA2. Subsequent references to the logical name CHARLIE result in the correspondence between the logical name CHARLIE and the disk and directory specified. Thus, the PRINT command queues a copy of the file DBA2:[CHARLES]TEST.DAT to the system printer.

2.
```
$ ASSIGN DBA1:  TEMP:
$ SHOW LOGICAL TEMP
  TEMP  =  DBA1:     (Process)
$ DEASSIGN TEMP
```

The ASSIGN command equates the logical name TEMP to the device DBA1. The SHOW LOGICAL command verifies that the logical name assignment was made. Note that the logical name TEMP was terminated with a colon in the ASSIGN command, but that that command interpreter deleted the colon before placing the name in the logical name table.

3.
```
$ MOUNT  MTB3:  MASTER  TAPE
$ ASSIGN  TAPE:NAMES.DAT  PAYROLL
$ RUN  PAY
    .
    .
```

The MOUNT command establishes the logical name TAPE for the device MTB3 which has the volume MASTER mounted on it. The ASSIGN command equates the logical name PAYROLL with the file named NAMES.DAT on the logical device TAPE. Thus, a subsequent OPEN request in a program that refers to the logical name PAYROLL results in the correspondence between the logical name PAYROLL and the file NAMES.DAT on the tape whose volume label is MASTER.

4.
```
$ ASSIGN/GROUP  _DBB1:  GROUP_DISK:
```

The ASSIGN command assigns the logical name GROUP_DISK to the physical device DBB1. Subsequently, another user in the same group can issue the command:

```
$ ASSIGN  GROUP_DISK:[HIGGINS]WEEKLY.OUT  OUTFILE
```

This ASSIGN command equates the logical name OUTFILE to a file on the device specified by the logical name GROUP_DISK. When the ASSIGN command executes, it locates the logical name GROUP_DISK in the group logical name table and translates it to the device name DBB1.

5.
```
$ ASSIGN/PROCESS/GROUP  DBA1:  SYSFILES:
$ SHOW LOGICAL  SYSFILES
  SYSFILES  =  DMA1:     (Group)
```

The ASSIGN command contains conflicting qualifiers. The response from the SHOW LOGICAL command indicates that the name was placed in the group logical name table.

6.
```
$ ASSIGN/GROUP 'F$LOGICAL("SYS$COMMAND") TERMINAL
  PREVIOUS LOGICAL NAME ASSIGNMENT REPLACED
```

The ASSIGN command uses the lexical function F$LOGICAL to translate the logical name SYS$COMMAND and use the result as the equivalence name for the logical name TERMINAL. The message from the ASSIGN command indicates an entry for the logical name TERMINAL already existed in the group logical name table, and that the new entry replaced the previous one.

If this command is used in a LOGIN.COM file, the entry for TERMINAL will be redefined at the beginning of each terminal session; the current process and any subprocesses it creates can execute images that use the logical name TERMINAL to write messages to the current terminal device.

7.
```
$ ASSIGN/USER_MODE SYS$COMMAND:  SYS$INPUT:
$ EDIT AVERAGE.FOR
$ FORTRAN AVERAGE
$ LINK AVERAGE
$ RUN AVERAGE
55
55
9999
```

In the command procedure illustrated above, the ASSIGN command equates the logical name SYS$INPUT with SYS$COMMAND for the execution of the next command. When the SOS editor is invoked, it will read all input from the current terminal, regardless of the number of active command levels. When the edit session is terminated, the deassignment is automatically made, and the procedure goes on to compile, link, and run the program AVERAGE.

The program AVERAGE reads input from the current default input device: in this example, test data records follow the RUN command in the input stream. Note that, if the assignment of SYS$INPUT were not made in user mode, there would be no way to re-establish the default relationship between the current input stream and the logical device named SYS$INPUT.

BASIC

Invokes the PDP-11 BASIC-PLUS-2[1] compiler to begin a BASIC session. All subsequent command input is read by BASIC-PLUS-2.

Format

BASIC

Command Qualifiers

None.

Prompts

Basic2

Description

After invoking BASIC-PLUS-2, use BASIC subcommands to create, edit, and compile BASIC programs.

To link and run a BASIC program, you must issue the BUILD subcommand to request BASIC to create a command procedure suitable for input to the RSX-11M Task Builder. After exiting from BASIC with the EXIT subcommand, create the executable image file by specifying the command procedure as input to the Task Builder.

For details on how to use BASIC-PLUS-2, see the BASIC-PLUS-2 RSX-11M/IAS User's Guide.

Examples

1.
```
$ BASIC

Basic Plus 2 V01.05

Basic2

OLD AVERAG

Basic2

COMPILE

Basic2

BUILD/SEQUENTIAL

Basic2

EXIT

$ MCR TKB @AVERAG

$ RUN AVERAG
```

1 Available under separate license

The BASIC command invokes BASIC-PLUS-2. The OLD, COMPILE, and BUILD subcommands define the input file, AVERAG.B2S, and request BASIC to compile the file and to create a command procedure for input to the task builder. The BUILD command creates the file AVERAG.CMD.

The MCR TKB command invokes the Task Builder to create an executable image file, using the commands in the file AVERAG.CMD. The RUN command executes the image AVERAG.EXE created by the Task Builder.

2.
```
$ CREATE CPYFIL.B2S
  .
  .
$ BASIC
OLD CPYFIL
COMPILE
BUILD/SEQUENTIAL
EXIT
$ MCR TKB @CPYFIL
$ ASSIGN TEST.DAT INFILE
$ ASSIGN TEST.OUT OUTFILE
$ RUN CPYFIL
$ TYPE TEST.OUT
```

This command procedure uses the CREATE command to create the BASIC source file, assigning it the name of CPYFIL.B2S. When the BASIC command executes, it reads subsequent input from the command input stream. After BASIC executes the OLD, COMPILE, and BUILD subcommands, the EXIT command terminates the BASIC session; the next commands are read by the DCL command interpreter.

After the MCR TKB command creates the image file, the ASSIGN commands assign equivalence names to the logical file names INFILE and OUTFILE. (These files must be referred to in BASIC OPEN statements in the file CPYFIL.EXE). The RUN command executes the image CPYFIL.EXE and the TYPE command verifies the program output in the file TEST.OUT.

The command procedure can be created interactively and submitted for execution as a batch job with the SUBMIT command, or punched on cards and submitted to a system card reader preceded with cards containing JOB and PASSWORD commands.

CANCEL

Cancels scheduled wakeup requests for a specified process. This includes wakeups scheduled with the RUN command and with the Schedule Wakeup ($SCHDWK) system service.

Format

```
CANCEL [process-name]

   Command Qualifiers

   /IDENTIFICATION=process-id
```

Prompts

None.

Command Parameters

process-name
Specifies the 1- to 15-alphanumeric character string name assigned to the process for which wakeup requests are to be canceled. Process names are generally assigned to processes when they are created. The specified process must have the same group number in its user identification code (UIC) as the current process.

If you also specify the /IDENTIFICATION qualifier, the process name is ignored. If you specify neither the process name parameter nor the /IDENTIFICATION qualifier, the CANCEL command cancels scheduled wakeup requests for the current process.

Description

The user privilege GROUP is required to cancel scheduled wakeups for non-owned processes in the same group; the user privilege WORLD is required to cancel scheduled wakeups for any process in the system.

The CANCEL command does not delete the specified process. If the process is executing an image when the CANCEL command is issued for it, the process hibernates instead of exiting after the image completes execution.

To delete a process that is hibernating and for which wakeup requests have been canceled, use the STOP command. You can determine whether a subprocess has been deleted by issuing the SHOW PROCESS command with the /SUBPROCESSES qualifier.

Command Qualifiers

/IDENTIFICATION=process-id
Specifies the process identification number the system assigned to the process when the process was created. When you specify the process identification, you can omit leading zeroes.

Examples

1.
```
$ RUN/SCHEDULE=14:00 STATUS
%RUN-S-PROC_ID, identification of created process is 0013012A
   .
   .
   .
$ CANCEL/IDENTIFICATION=13012A
```

THE RUN command creates a process to execute the image STATUS. The process hibernates, and is scheduled to be awakened at 14:00. Before the process is awakened, the CANCEL command cancels the wakeup request.

2.
```
$ RUN/PROCESS_NAME=LIBRA/INTERVAL=1:00 LIBRA
%RUN-S-PROC_ID, identification of created process is 00130027
    .
    .
    .
$ CANCEL LIBRA
$ STOP LIBRA
```

The RUN command creates a subprocess named LIBRA to execute the image LIBRA.EXE at hourly intervals.

Subsequently, the CANCEL command cancels the wakeup requests. The process continues to exist, but in a state of hibernation. The STOP command deletes the subprocess.

CLOSE

Closes a file that was opened for input or output with the OPEN command and deassigns the logical name specified when the file was opened.

Format

```
CLOSE  logical-name

   Command Qualifiers

   /ERROR=label
```

Prompts

Log_Name: logical-name

Command Parameters

logical-name
Specifies the logical name to be assigned to the file when it was opened with the OPEN command.

Description

Files that are opened for reading or writing at the command level remain open until explicitly closed with the CLOSE command, or until the process is deleted at logout. If a command procedure that opens a file terminates without closing an open file, the file remains open; the command interpreter does not automatically close it.

Command Qualifiers

/ERROR=label
Specifies a label on a line in the command procedure to receive control if the close request results in an error. If no error label is specified and an error occurs during the closing of the file, the command procedure continues execution at the next line in the file, as it does if no error occurs.

The error routine specified for this qualifier takes precedence over any action statement indicated in an ON command. If /ERROR is not specified, the current ON condition action is taken.

If an error occurs and the target label is successfully given control, the global symbol $STATUS contains a successful completion value.

Examples

1.
```
$ OPEN/READ INPUT_FILE  TEST.DAT
$ READ_LOOP:
$ READ/END_OF_FILE=NO_MORE  INPUT_FILE  DATA_LINE
    .
    .
$ GOTO READ_LOOP
$ NO_MORE :
$ CLOSE INPUT_FILE
```

The OPEN command opens the file TEST.DAT and assigns it the logical name of INPUT_FILE. The /END_OF_FILE qualifier on the READ command requests that when the end of file is reached, the command interpreter should transfer control to the line at the label NO_MORE. The CLOSE command closes the input file.

2.
```
$ @READFILE
^Y
$ STOP
$ SHOW LOGICAL/PROCESS
    .
    .
  INFILE = _DB1
  OUTFILE = _DB1
$ CLOSE INFILE
$ CLOSE OUTFILE
```

CTRL/Y interrupts the execution of the command procedure READFILE.COM and the STOP command stops it. The SHOW LOGICAL/PROCESS command displays the names that currently exist in the process logical name table. Among the names listed are the logical names INFILE and OUTFILE, assigned by OPEN commands in the procedure READFILE.COM.

THE CLOSE commands close these files.

COBOL/RSX11

Invokes the PDP-11 COBOL-74/VAX[1] compiler to compile a COBOL source program. The /RSX11 qualifier is required.

Format

```
COBOL/RSX11    file-spec

   Command Qualifiers          Default

   /[NO]ANSI_FORMAT            /NOANSI_FORMAT
   /[NO]COPY_LIST              /NOCOPY_LIST
   /[NO]CROSS_REFERENCE        /NOCROSS_REFERENCE
   /[NO]LIST[=file-spec]
   /[NO]MAP                    /NOMAP
   /NAMES=aa                   /NAMES=$C
   /NEST=n                     /NEST=10
   /[NO]OBJECT[=file-spec]
   /[NO]OVERLAY                /NOOVERLAY
   /RSX11
   /SEGMENT_SIZE=n
   /[NO]VERB_LOCATION          /NOVERB_LOCATION
   /[NO]WARNINGS               /NOWARNINGS
```

Prompts

```
File:      file-spec
```

Command Parameters

file-spec
: Specifies the COBOL source program to be compiled. The file specification must contain a file name; if you do not specify a file type, the compiler uses the default file type of CBL.

  Wild cards are not allowed in the file specification.

Command Qualifiers

/ANSI_FORMAT
/NOANSI_FORMAT
: Indicates whether the source program is in ANSI COBOL format or in DIGITAL's Terminal format.

  An ANSI COBOL source file has 80-character records with Area A beginning in record position 8.

  By default, the COBOL/RSX11 command assumes that the input records are in Terminal format, that is, Area A begins in record position 1 and the records do not have line numbers.

---

[1] Available under separate license

/COPY_LIST
/NOCOPY_LIST
Controls whether statements produced by COPY statements in the source program are printed in the listing file.

/COPY_LIST is the default: all source statements are included in the output listing.

/CROSS_REFERENCE
/NOCROSS_REFERENCE
Controls whether the compiler listing includes a cross reference listing. By default, the compiler does not create a cross reference listing.

/LIST[=file-spec]
/NOLIST
Controls whether the compiler produces an output listing and defines characteristics of the file.

If you issue the COBOL/RSX11 command from interactive mode, the compiler, by default, does not create a listing file. If you specify /LIST without a file specification, the compiler creates a listing with the same file name as the input file, but uses the file type of LST. If you include a file specification, the listing is written to that file or device.

If the COBOL/RSX11 command is executed from a batch job, /LIST is the default.

/MAP
/NOMAP
Requests the compiler to produce a Data Division map showing the memory addresses for Data Division entries.

/NOMAP is the default: the compiler does not produce a map.

/NAMES=aa
Requests the compiler to generate PSECT names starting with the 2-character prefix aa.

If the /NAMES qualifier is not specified, the default prefix of $C is used.

/NEST=n
Specifies the number of nested PERFORM statements allowed in the source program. By default, the compiler allows a maximum of 10 nested PERFORM statements.

/OBJECT[=file-spec]
/NOOBJECT
Controls whether the compiler produces an object file.

By default, the compiler produces an object file with the same file name as the input file and a file type of OBJ. The compiler also uses the default file type of OBJ when you include a file specification with the /OBJECT qualifier that does not have a file type.

/OVERLAY
/NOOVERLAY
Controls whether the compiler makes procedural PSECTs overlayable.

/RSX11
Requests the PDP-11 COBOL/74-VAX compiler.

/SEGMENT_SIZE=n
Specifies the maximum size, in bytes, of procedure PSECTs created by the compiler. The minimum value allowed for n is 108 bytes.

/VERB_LOCATION
/NOVERB_LOCATION
Indicates whether the output listing produced by the compiler shows the object location of each verb in the source program.

/WARNINGS
/NOWARNINGS
Controls whether the compiler prints informational diagnostic messages as well as warning and fatal diagnostic messages. By default, the compiler prints informational diagnostics; specify /NOWARNINGS to suppress them.

Examples

1. `$ COBOL/RSX11  MYFILE`

   The COBOL command compiles the source statements in the file MYFILE.CBL and produces an object file named MYFILE.OBJ.

2. `$ COBOL/RSX11/OBJECT=TEST2/LIST  TEST`

   The COBOL command compiles the source statements in the file TEST.CBL and produces an object file named TEST2.OBJ and a listing file named TEST.LST.

3.
```
$ COBOL/RSX11 SCANLINE
$ RUN SYS$SYSTEM:MRG
PLEASE ENTER FILE SPECIFICATION FOR OUTPUT FILE
SCAN.ODL
DO YOU WANT AN ABBREVIATED OR MERGED ODL FILE?
PLEASE ANSWER A(BBREVIATED) OR M(ERGED) A
DO YOU WANT TO OVERLAY I/O SUPPORT ROUTINES?
PLEASE ANSWER Y(ES) OR N(O) N
PLEASE ENTER FILE SPECIFICATION FOR INPUT ODL FILE
SCANLINE
OBJECT PROGRAM REFERENCED IN ODL FILE IS:
     SCANLINE.OBJ
PLEASE ENTER OBJECT FILE DEVICE AND UIC IN THE FORMAT: DEV:GROUP,MEMBER]
(RET)
ANY MORE INPUT ODL FILES?
PLEASE ANSWER Y(ES) OR N(O) N
ODL FILE MERGE COMPLETE
MERGED ODL FILE IS:
SCAN.ODL

CBL -- 15: STOP RUN

$ LINK/RSX11 SCAN/OVERLAY
$ RUN SCAN
```

   The COBOL/RSX11 command compiles the source program SCANLINE.CBL. The RUN command invokes the COBOL MERGE utility to create an overlay description file. The output from the MERGE utility, SCAN.ODL, is then specified as input to the LINK/RSX11 command. This command creates the executable image file, SCAN.EXE. The RUN command executes the image.

CONTINUE

Resumes execution of a DCL command, a program, or a command procedure that was interrupted by pressing CTRL/Y or CTRL/C. The CONTINUE command can also serve as the target command of an IF or ON command in a command procedure, or following a label that is the target of a GOTO command.

You can truncate the CONTINUE command to a single letter, C.

Format

```
CONTINUE
```

Command Qualifiers

None.

Prompts

None.

Command Parameters

None.

Description

After you interrupt an image, you cannot continue its execution if you have entered any command that executes another image. For a list of the commands that do not execute separate images, see Section 4.2.3, "Interrupting Program Execution."

Examples

1.
```
$ RUN  MYPROGA
^Y
$ SHOW TIME
  18-JAN-1978 13:40:12
$ CONTINUE
```

The RUN command executes the program MYPROG. While the program is running, pressing CTRL/Y interrupts the image. The SHOW TIME command requests a display of the current date and time. The CONTINUE command resumes the image.

2.
```
$ ON SEVERE_ERROR THEN CONTINUE
```

This statement in a command procedure requests the command interpreter to continue executing the procedure if any warning, error, or severe error status value is returned from the execution of a command or program. This ON statement overrides the default action, which is to exit from a procedure following errors or severe errors.

COPY

Creates a new file from one or more existing files. The COPY command can:

- Copy one file to another file
- Concatenate more than one file into a single output file
- Copy a group of files to another group of files

Format

```
COPY    input-file-spec,...   output-file-spec

   Command Qualifiers        Default

   /[NO]CONCATENATE          /CONCATENATE
   /[NO]LOG                  /NOLOG


   File Qualifiers

   /ALLOCATION=n
   /[NO]CONTIGUOUS           /NOCONTIGUOUS
   /EXTENSION=n
   /FILE_MAXIMUM=n
   /[NO]OVERLAY              /NOOVERLAY
   /PROTECTION=code
   /[NO]READ_CHECK           /NOREAD_CHECK
   /[NO]REPLACE              /NOREPLACE
   /[NO]TRUNCATE             /NOTRUNCATE
   /[NO]WRITE_CHECK          /NOWRITE_CHECK
```

Prompts

From: input-file-spec,...

To: output-file-spec

Command Parameters

input-file-spec,...
Specifies the names of one or more input files to be copied. If you specify more than one input file, you can separate them with either commas (,) or plus signs (+).

You can use wild cards (*) in place of the file name, type, or version fields. Then, all files that satisfy the remaining components are copied.

output-file-spec
Specifies the name of the output file into which the input files are to be copied.

You must specify at least one field in the output file specification. If you do not specify a device and/or directory, the COPY command uses your current default device and directory. For other fields that you do not specify, the COPY command uses the corresponding field of the input file specification.

If you specify a wild card in place of the file name, type, and/or version field of the output file specification, the COPY command creates one or more output files, based on the input file specification. It uses the corresponding field of the first or related input file specification to name the output file.

Description

When you specify more than one input file the COPY command creates, by default, a single output file. You can specify multiple input files in any of the following ways:

- Separate input file specifications with commas (,) or plus signs (+).
- Specify a wild card in place of the file name, type, and/or version field of an input file specification.

The COPY command creates multiple output files when you specify multiple input files and one of the following:

- An explicit wild card in the output file name, type, and/or version field.
- Only a device and/or directory name in the output file specification.
- The /NOCONCATENATE qualifier.

When the COPY command creates multiple output files, it uses the corresponding field of each input file to name an output file.

When the COPY command creates a single output file for which any field of the output file specification contains a wild card, the COPY command uses the corresponding field of the first, or only, input file to name the output file.

Use the /LOG qualifier when you specify multiple input and output files to verify the files actually copied.

Version Numbers: If no version numbers are specified for input and output files, the COPY command, by default, gives the output file a version number of 1, or increments by 1 the version number of an existing file with the same file name and file type. If explicit version numbers or wild cards are specified for input files, the COPY command, by default, gives the output files the same version numbers as the associated input files. In the latter case, if an equal or higher version of the output file already exists, the COPY command issues a warning message and does not copy the file.

Command Qualifiers

/CONCATENATE
/NOCONCATENATE

Controls, when a wild card is used in any component of the output file specification, whether a single output file is to be created from all files that satisfy the input file specification.

By default, a wild card in an input file specification results in a single output file consisting of multiple files that match in one or more fields of the file specification.

When you concatenate files from On-disk Structure Level 2 disks, the COPY command concatenates the files in alphanumeric order; if you specify a wild card in the file version field, files are copied in descending order by version number. When you concatenate files from On-disk Structure Level 1 disks, the COPY command concatenates the files in random order.

/LOG
/NOLOG

Controls whether the COPY command displays the file specifications of each file copied.

If you specify /LOG, the COPY command displays, for each copy operation, the file specifications of the input and output files, the number of blocks or the number of records copied (depending on whether the file is copied on a block-by-block or record-by-record basis), and the total number of new files created.

File Qualifiers

/ALLOCATION=n

Forces the initial allocation of the output file to the number of 512-byte blocks specified as n.

If not specified, the initial allocation of the output file is determined by the size of the input file being copied.

/CONTIGUOUS
/NOCONTIGUOUS

Indicates whether the output file is to be contiguous, that is, whether the file must occupy consecutive physical disk blocks.

By default, the COPY command creates an output file in the same format as the corresponding input file. If an input file is contiguous, the COPY command attempts to create a contiguous output file, but it does not report an error if there is not enough space. If you copy multiple input files of different formats, the output file may or may not be contiguous. Use the /CONTIGUOUS qualifier to ensure that files are copied contiguously.

The /CONTIGUOUS qualifier has no effect when you copy files to or from tapes because the size of the input file cannot be determined. If you copy a file from a tape and want the file to be contiguous, use two COPY commands: once, to copy the file from the tape, and a second time to create a contiguous file.

/EXTENSION=n
Specifies the number of blocks to be added to the output file each time the file is extended.

If you do not specify /EXTENSION, the default extension attribute of the output file is determined by the extension attribute of the corresponding input file.

/FILE_MAXIMUM=n
Specifies the maximum number of logical records that the output file can contain.

This qualifier is valid only for relative files. For information on creating and using relative files, see the VAX-11 Record Management Services Reference Manual.

/OVERLAY
/NOOVERLAY
Requests that data in the input file be copied into an existing output file, overlaying the existing data. The physical location of the file on disk does not change.

The /OVERLAY qualifier is ignored if the output file is written to a non-file-structured device.

/PROTECTION=code
Defines the protection to be applied to the output file.

Specify the protection code using the standard rules for specifying protection (these rules are given in the discussion of the SET PROTECTION command). Any protection attributes not specified are taken from the current protection of the corresponding input file.

/READ_CHECK
/NOREAD_CHECK
Requests the COPY command to read each record in the specified input file(s) twice to verify that all records were correctly read.

/REPLACE
/NOREPLACE
Requests that if a file already exists with the same file specification as that entered for the output file, the existing file is to be deleted. The COPY command allocates new space for the output file.

By default, the COPY command creates a new version of a file if the file already exists, incrementing the version number.

/TRUNCATE
/NOTRUNCATE
Controls whether the COPY command truncates an output file at the end-of-file when copying it. By default, the COPY command uses the allocation of the input file to determine the size of the output file.

/WRITE_CHECK
/NOWRITE_CHECK
Requests the COPY command to read each record in the output file after it was written to verify that the record was successfully copied and that the file can subsequently be read without error.

Examples

1.
```
$ COPY   TEST.DAT   NEWTEST.DAT
```

The COPY command copies the contents of the file TEST.DAT from the default disk and directory into a file named NEWTEST.DAT. If a file named NEWTEST.DAT already exists, the COPY command creates a new version of it.

2.
```
$ COPY  ALPHA.TXT  TMP
$ COPY  ALPHA.TXT  .TMP
```

The first COPY command copies the file ALPHA.TXT into a file named TMP.TXT. The COPY command uses the file type of the input file to complete the file specification for the output file. The second COPY command creates a file named ALPHA.TMP. The COPY command uses the file name of the input file to name the output file.

3.
```
$ COPY/LOG/REPLACE   TEST.DAT  NEWTEST.DAT;1
%COPY-I-REPLACED, DBA2:[MALCOLM]NEWTEST.DAT;1 being replaced
%COPY-S-COPIED, DBA2:[MALCOLM]TEST.DAT;1 copied to DBA2:[MALCOLM]NEWTEST.DAT;1 (1 block)
%COPY-S-NEWFILES, 1 file created
```

The /REPLACE qualifier requests the COPY command to replace an existing version of the output file with the new file. The first message from the COPY command indicates that it is replacing an existing file. The version number in the output file must be explicit; otherwise, the COPY command creates a new version of the file NEWTEST.DAT.

4.
```
$ COPY  *.COM   [MALCOLM.TESTFILES]
```

The COPY command copies the highest versions of files in the current default directory with a file type of COM to the subdirectory MALCOM.TESTFILES.

5.
```
$ COPY/LOG *.TXT *.OLD
%COPY-S-COPIED, DBA2:[MALCOLM]ALPHA.TXT;2 copied to DBA2:[MALCOLM]ALPHA.OLD;2 (1 block)
%COPY-S-COPIED, DBA2:[MALCOLM]BETA.TXT;2 copied to DBA2:[MALCOLM]BETA.OLD;2 (1 block)
%COPY-S-COPIED, DBA2:[MALCOLM]GAMMA.TXT;2 copied to DBA2:[MALCOLM]GAMMA.OLD;2 (4 blocks)
%COPY-S-NEWFILES, 3 files created
```

The COPY command copies the highest versions of files with file types of TXT into new files. Each new file has the same file name as an existing file, but a file type of OLD. The last message from the COPY command indicates the number of new files that it created.

6.
```
$ COPY/LOG   A.DAT,B.MEM   C.*
%COPY-S-COPIED, DBA2:[MALCOLM]A.DAT;5 copied to DBA2:[MALCOLM]C.DAT;11 (1 block)
%COPY-S-COPIED, DBA2:[MALCOLM]B.MEM;2 copied to DBA2:[MALCOLM]C.MEM;24 (58 records)
%COPY-S-NEWFILES, 2 files created
```

The input file specifications are separated with a comma to request that two files be copied. The wild card in the output file specification indicates that two output files are to be created. For each copy operation, the COPY command uses the file type of the input file to name the output file.

7.

```
$ COPY/LOG   *.TXT   TXT.SAV
%COPY-S-COPIED, DBA2:[MALCOLM]ALPHA.TXT;2 copied to DBA2:[MALCOLM]TXT.SAV;1 (1 block)
%COPY-S-APPENDED, DBA2:[MALCOLM]BETA.TXT;2 appended to DBA2:[MALCOLM]TXT.SAV;1 (3 records)
%COPY-S-APPENDED, DBA2:[MALCOLM]GAMMA.TXT;2 appended to DBA2:[MALCOLM]TXT.SAV;1 (51 records)
%COPY-S-NEWFILES, 1 file created
```

The COPY command copies the highest versions of all files with file types of TXT to a single output file named TXT.SAV. After the first input file is copied, the messages from the COPY command indicate that subsequent files are being appended to the output file.

Note that if you specify /NOCONCATENATE in this example, the COPY command creates multiple versions of the file TXT.SAV.

8.
```
$ COPY MASTER.DOC DMA1:[BACKUP]
```

The COPY command copies the highest version of the file MASTER.DOC to the device DMA1. If no file named MASTER.DOC already exists in the directory BACKUP, the COPY command uses the version number of the input file.

9.
```
$ MOUNT   MTA1:   VOL025   TAPE:
$ COPY  TAPE:*.*   *
```

The MOUNT command requests that the volume labeled VOL025 be mounted on the tape device MTA1 and assigns the logical name TAPE to the device.

The COPY command uses the logical name TAPE for the input file specification, requesting that all files on the tape be copied to the current default disk and directory. All the files copied retain their file names and file types.

10.
```
$ ALLOCATE CR:
  _CRA0:  ALLOCATED
$ COPY CRA0:  CARDS.DAT
$ DEALLOCATE CRA0:
```

The ALLOCATE command allocates a system card reader for exclusive use by the process. The response from the ALLOCATE command indicates the device name of the card reader, CRA0.

After the card reader is allocated, you can place a deck of cards in the reader and issue the COPY command specifying the card reader as the input file. The COPY command reads the cards into the file CARDS.DAT. The end-of-file in the card deck must be indicated with an EOF card (12-11-0-1-6-7-8-9 overpunch).

The DEALLOCATE command relinquishes use of the card reader.

CREATE

Creates a sequential disk file from records that follow the command in the input stream, or creates a directory file.

Format

```
CREATE   file-spec

   Command Qualifiers        Default

   /DIRECTORY
   /OWNER_UIC=uic
   /PROTECTION=code
```

Prompts

File: file-spec

Command Parameters

file-spec
Specifies the name of the input file or directory to be created.

If you omit either the file name or the file type, the CREATE command does not supply any defaults; the file name or file type is null. If you do not specify a file version number, and a file already exists with the same file name and file type as the file specification, the CREATE command creates a new version of the file.

No wild cards are allowed in the file specification.

If you specify /DIRECTORY, the file specification must contain a directory name, and optionally can contain a device name. When you create a subdirectory, separate the names of the directory levels with periods (.).

Command Qualifiers

/DIRECTORY
Indicates that a directory or subdirectory is to be created.

To create a first level directory, you must be allowed write access to the master file directory on the volume on which you are creating the directory. On a system volume, you must have a system UIC to create a first level directory. To create a subdirectory, you must be allowed write access to the first level directory or have an appropriate user privilege.

/OWNER_UIC=uic

Specifies the user identification code to be associated with the directory being created. Specify the UIC in the format:

[g,m]

g is an octal number in range 0-377 representing the group number.

m is an octal number in the range 0-377 representing the member number.

The brackets are required.

If you do not specify an owner UIC when you create a subdirectory, the command uses the owner UIC of the next highest level directory.

/PROTECTION=code

Defines the protection to be applied to the file or directory.

Specify the code according to the standard syntax rules for specifying protection. These rules are given in the description of the SET PROTECTION command. Any attributes not specified are taken from the current default protection.

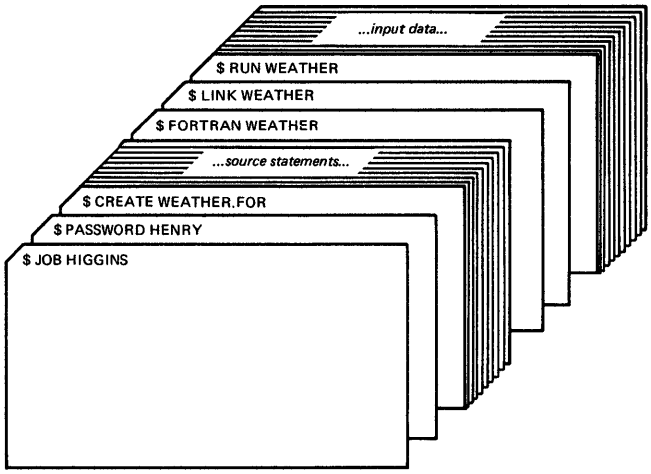
If you do not specify a protection when you create a subdirectory, the command uses the protection in effect for the next highest level directory. The default protection for directory files on system volumes is read, write, and execute.

Examples

1.
```
$ CREATE   A.DAT
Input line one...
Input line two...
    .
    .
^Z
$
```

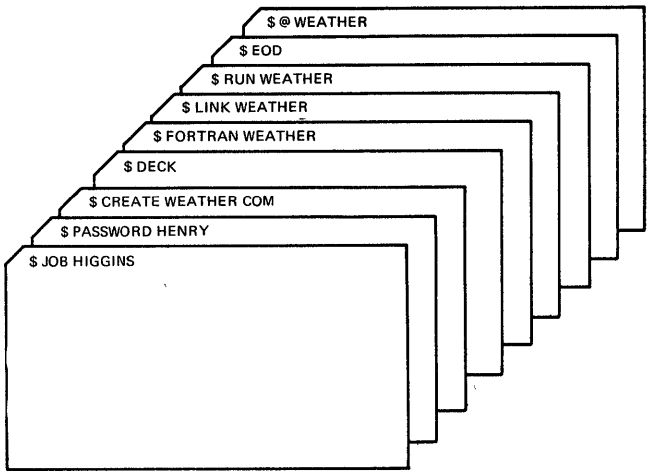
After you issue the CREATE command from the terminal, the system reads input lines into the sequential file A.DAT until CTRL/Z terminates the input.

2.



When you issue the CREATE command from a command procedure file, the system reads all subsequent lines in the command procedure file into the file, until it encounters a dollar sign ($) in the first position in a record. In this batch job example, the CREATE command creates a FORTRAN source file. The next commands compile, link, and run the file just created. Input data follows the RUN command.

3.



This batch job example illustrates using the CREATE command to create a command procedure from data in the input stream. The DECK command is required so that subsequent lines that begin with a dollar sign are not executed as commands, but are accepted as input records. The EOD command signals the end-of-file for the data records. Then, the procedure is executed with the @ (Execute Procedure) command.

4.
```
$ CREATE/DIRECTORY   DMA2:[MALCOLM]
```

The CREATE command creates a directory named MALCOLM on the device DMA2.

5.
```
$ CREATE/DIRECTORY   [MALCOLM.SUB]
$ SET DEFAULT [MALCOLM.SUB]
```

The CREATE command creates a subdirectory named MALCOLM.SUB. This directory file is placed in the directory named MALCOLM. The SET DEFAULT command changes the current default directory to this subdirectory. All files subsequently created are cataloged in MALCOLM.SUB.

DEALLOCATE

Returns a device that was reserved for private use to the pool of available devices in the system.

Format

```
DEALLOCATE   [device-name[:]]

   Command Qualifiers

   /ALL
```

Prompts

Device: device-name

Command Parameters

device-name
Specifies the name of the device to be deallocated. The device name can be a physical device name or a logical name.

If you omit the controller designator and/or unit number, they default to controller A and unit 0.

Command Qualifiers

/ALL
Requests that all devices you have currently allocated be deallocated.

If you specify /ALL, you cannot specify a device name.

Examples

1.
```
$ DEALLOCATE  _DMB1:
```

The DEALLOCATE command deallocates unit 1 of the RK06 device on controller B. The underscore character in the device name indicates that it is a physical device name; the DEALLOCATE command does not check to see if it is a logical name.

2.
```
$ ALLOCATE   MT:   TAPE:
  _MTB1:  ALLOCATED
    .
    .
$ DEALLOCATE  TAPE
```

The ALLOCATE command requests that any magnetic tape drive be allocated and assigns the logical name TAPE to the device. The response to the ALLOCATE command indicates the successful allocation of the device MTB1. The DEALLOCATE command specifies the logical name TAPE to release the tape.

Note that a colon was specified on the logical name TAPE in the ALLOCATE command, but that the colon can be omitted on the DEALLOCATE command.

3. `$ DEALLOCATE/ALL`

   The DEALLOCATE command deallocates all devices that are currently allocated.

DEASSIGN

Cancels logical name assignments made with the ASSIGN, DEFINE, ALLOCATE, or MOUNT commands.

Format

```
DEASSIGN   [logical-name[:]]

   Command Qualifiers        Default

   /ALL
   /GROUP                    /PROCESS
   /PROCESS                  /PROCESS
   /SUPERVISOR_MODE          /SUPERVISOR_MODE
   /SYSTEM                   /PROCESS
   /USER_MODE                /SUPERVISOR_MODE
```

Prompts

Log_Name: logical-name

Command Parameters

logical-name

Specifies a 1- to 63-character logical name to be deassigned. If the logical name contains any characters other than alphanumeric or underscore characters, enclose it in quotation marks (").

If you terminate the logical-name parameter with a colon (:), the command interpreter ignores it. (Note that the ASSIGN, ALLOCATE, and MOUNT commands remove a trailing colon, if present, from a logical name before placing the name in a logical name table.) If a colon is present in the actual logical name, you must specify two colons on the logical-name parameter for the DEASSIGN command.

The logical-name parameter is required unless you specify /ALL.

Description

If you enter more than one of the qualifiers /PROCESS, /GROUP, or /SYSTEM, only the last one entered is accepted. If entries exist for the specified logical name in more than one logical name table, the name is deleted only from the specified logical name table.

The command interpreter deassigns all supervisor mode entries in the process logical name table when you log off the system. User mode entries are deassigned when any image exits. Names in the group or system logical name tables must be explicitly deassigned.

Command Qualifiers

/ALL
Specifies that all logical names in the specified logical name table are to be deleted. If no logical name table is specified, all process logical name table entries are deleted.

If you specify /ALL, you cannot enter a logical-name parameter.

/GROUP
Indicates that the specified logical name is in the group logical name table.

The user privilege GRPNAM is required to delete entries from the group logical name table.

/PROCESS
Indicates that the specified logical name is in the process logical name table. This is the default.

You cannot deassign logical name table entries that were made by the command interpreter, for example SYS$INPUT, SYS$OUTPUT, and SYS$ERROR. However, if you assign new equivalence names for these logical names, you can deassign the names you explicitly created.

/SUPERVISOR_MODE
Indicates, for entries in the process logical name table, that an entry exists in supervisor mode. This is the default; /SUPERVISOR_MODE deletes both user and supervisor mode entries.

/SYSTEM
Indicates that the specified logical name is in the system logical name table.

The user privilege SYSNAM is required to delete entries from the system logical name table.

/USER_MODE
Indicates, for entries in the process logical name table, that the entry exists in user mode. /USER_MODE deletes only user mode entries.

Examples

1.
```
$ SHOW LOGICAL TEST_CASES
  TEST_CASES = DBA1:[HARVEY]FILES.DAT (process)
$ DEASSIGN   TEST_CASES
$ SHOW LOGICAL TEST_CASES
  No translation for logical name TEST_CASES
```

The SHOW LOGICAL command displays the current equivalence name for the logical name TEST_CASES. The DEASSIGN command deassigns the equivalence name; the next SHOW LOGICAL command indicates that the name is deassigned.

2.
```
$ ASSIGN DBA1:   COPY:
$ DEASSIGN COPY
```

The ASSIGN command equates the logical name COPY with the device DBA1 and places the names in the process logical name table. The DEASSIGN command deletes the logical name. Note that a colon was specified on the logical name COPY in the ASSIGN command, but that the colon can be omitted on the DEASSIGN command.

3.
```
$ DEFINE SWITCH:  TEMP
$ DEASSIGN SWITCH::
```

   The DEFINE command places the logical name SWITCH: in the process logical name table. Two colons are required on the DEASSIGN command to delete this logical name because the DEFINE command does not remove trailing colons from logical names.

4.
```
$ ASSIGN/GROUP _DBB2:  GROUP_DISK
$ DEASSIGN/PROCESS/GROUP GROUP_DISK
```

   The ASSIGN command places the logical name GROUP_DISK in the group logical name table. A subsequent DEASSIGN command specifies conflicting qualifiers; because the /GROUP qualifier is last, the name is successfully deassigned.

5.
```
$ DEASSIGN/ALL
```

   The DEASSIGN command deletes all names from the process logical name table. This command does not, however, delete the names that were placed in the process logical name table in executive mode by the command interpreter (SYS$INPUT, SYS$OUTPUT, SYS$ERROR, SYS$DISK, and SYS$COMMAND).

DEBUG

Invokes the VAX-11 Symbolic Debugger after program execution is interrupted by CTRL/C or CTRL/Y. The program image being interrupted must contain the debugger, that is, the image was linked with the /DEBUG qualifier and/or run with the /DEBUG qualifier.

Format

```
DEBUG

   Command Qualifiers

   None.
```

Prompts

DBG>

Command Parameters

None.

Description

When a program image is executing, it can be interrupted by CTRL/C or CTRL/Y. Following the interruption, the DEBUG command can be issued to pass control to the debugger; this function is useful when a program is in an infinite loop and you want to gain control and use the debugger to determine the cause of the problem.

If no image is currently executing, the DEBUG command performs no operation.

If the interrupted program was not linked with the debugger, the DEBUG command causes a software exception condition. If the image has not declared a condition handler, this exception condition may cause the termination of the image.

For complete details on the commands available to debug programs, see the VAX-11 Symbolic Debugger Reference Manual.

Example

1.
```
$ FORTRAN/DEBUG/NOOPTIMIZE   WIDGET
$ LINK/DEBUG  WIDGET
$ RUN WIDGET

     DEBUG Version 1.0 24 Aug 1978
%DEBUG-I-INITIAL, language is FORTRAN, scope and  module  set
to 'WIDGET'
DBG>GO
ENTER NAME:
ENTER NAME:
ENTER NAME:
^Y
$ DEBUG
DBG>
```

The FORTRAN and LINK commands both specify the /DEBUG qualifier, to compile the program WIDGET with debugger symbol table information and to include the debugger in the image file. The RUN command begins execution of the image WIDGET, which loops uncontrollably. CTRL/Y interrupts the program, and the DEBUG command gives control to the debugger.

DECK

Marks the beginning of an input stream for a command or program. The DECK command is required in command procedures when the first non-blank character in any data record in the input stream is a dollar sign ($).

The DECK command must be preceded by a $; the $ must be in the first character position (column 1) of the input record.

Format

DECK

Command Qualifiers	Default
/DOLLARS[=string]	/DOLLARS=$EOD

Prompts

None.

Command Parameters

None.

Description

The DECK command defines an end-of-file indicator only for a single data stream; it allows you to place data records beginning with dollar signs in the input stream. You can place one or more sets of data in the input stream following a DECK command, each terminated by an end-of-file indicator.

After an end-of-file indicator specified with the /DOLLARS qualifier is encountered, the end-of-file indicator is reset to the default, that is, any record beginning with a dollar sign ($). The default is also reset if an actual end-of-file occurs for the current command level.

The DECK command is invalid if it is not preceded by a request to execute a command or program that requires input data.

Command Qualifiers

/DOLLARS[=string]

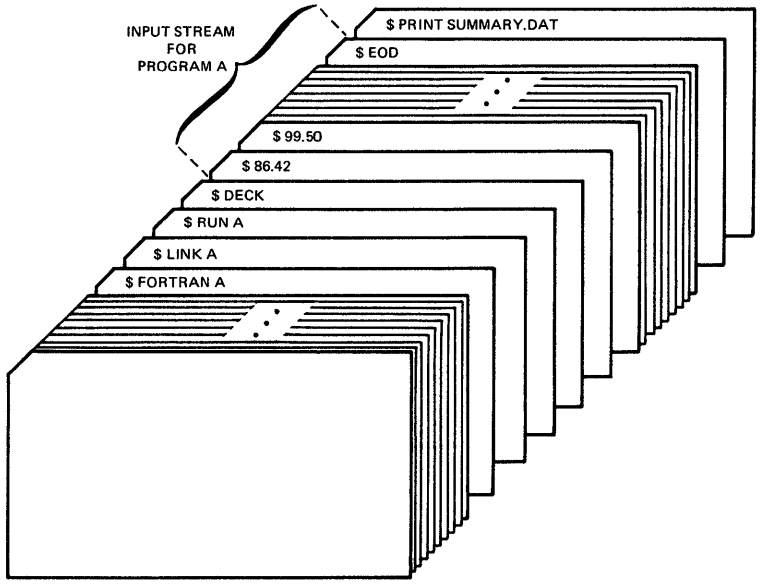
Sets the end-of-file indicator to the specified string.

If you do not specify /DOLLARS, or if you specify /DOLLARS without specifying a string, you must use the EOD command to signal the end-of-file.

Specify a string if the input data contains one or more records beginning with the string $EOD. The string can have from 1 to 15 characters. Enclose it in quotation marks if you want to specify an end-of-file indicator that contains literal lowercase letters or multiple blank spaces or tabs.

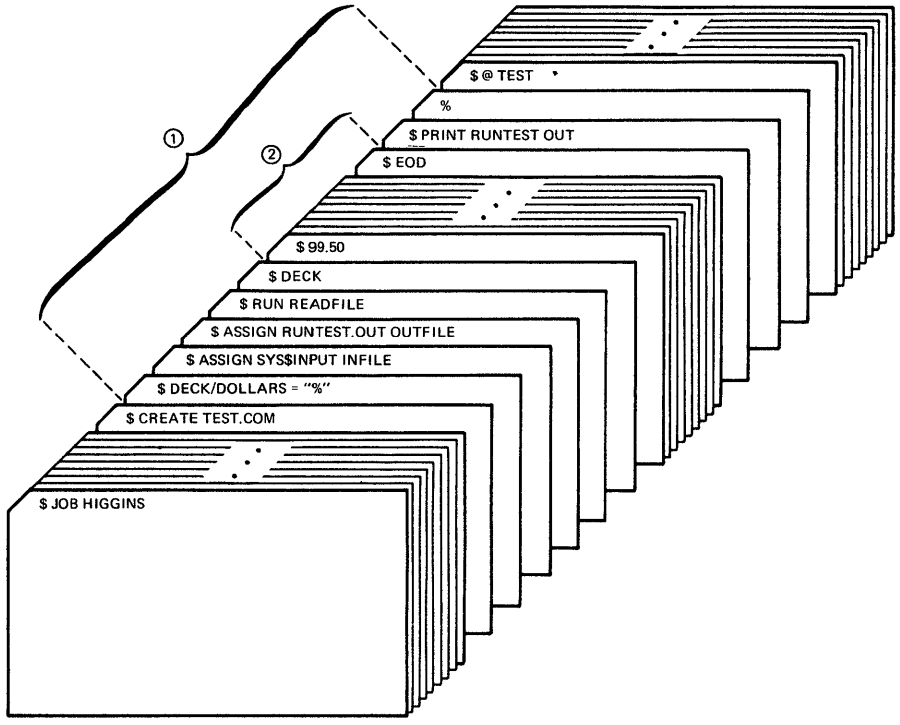
Examples

1.



The FORTRAN and LINK commands compile and link program A. When the program is run, any data the program reads from the logical device SYS$INPUT is read from the command stream. The $DECK command indicates that the input stream may contain dollar signs. The $EOD command signals end-of-file for the data.

2.



① INPUT STREAM FOR CREATE COMMAND

② INPUT STREAM FOR PROGRAM READFILE

The CREATE command creates the command procedure file TEST.COM from lines entered into the input stream. The $DECK/DOLLARS command indicates that the percent-sign (%) character is the end-of-file indicator for the CREATE command. This allows the string $ EOD to be read as an input record, signalling the end of the input for the RUN command.

DEFINE

Creates a logical name table entry and assigns an equivalence name string to the specified logical name. The DEFINE command is similar in function to the ASSIGN command; however, its primary purpose is to assign logical name/equivalence name pairs for application-specific uses other than for logical file specification assignments.

Format

```
DEFINE   logical-name  equivalence-name

   Command Qualifiers        Default

   /GROUP                    /PROCESS
   /PROCESS                  /PROCESS
   /SUPERVISOR_MODE          /SUPERVISOR_MODE
   /SYSTEM                   /PROCESS
   /USER_MODE                /SUPERVISOR_MODE
```

Prompts

Log_Name: logical-name

Equ_Name: equivalence-name

Command Parameters

logical-name
: Specifies a 1- to 63-character logical name string. If the string contains any characters besides alphanumeric or underscore characters, enclose it in quotation marks (").

equivalence-name
: Defines the 1- to 63-character equivalence name to be associated with the logical name in the specified logical name table. If the string contains other than alphanumeric or underscore characters, it must be enclosed in double quotation marks (").

Description

If you enter more than one of the qualifiers /PROCESS, /GROUP, or /SYSTEM, or both of the qualifiers /SUPERVISOR_MODE and /USER_MODE, only the last qualifier entered is accepted.

For additional information on using logical names, see Section 2.2, "Logical Names."

Command Qualifiers

/GROUP
Places the logical name/equivalence name pair in the group logical name table. Other users who have the same group number in their UICs (user identification codes) can access the logical name.

The user privilege GRPNAM is required to place a name in the group logical name table.

/PROCESS
Places the logical name/equivalence name pair in the process logical name table. This is the default.

/SUPERVISOR_MODE
Specifies, for an entry in the process logical name table, that the logical name be entered in supervisor mode.

This is the default for process logical name table entries. The /SUPERVISOR_MODE qualifier is ignored when entries are made in the group or system logical name tables.

/SYSTEM
Places the logical name/equivalence name pair in the system logical name table. All system users can access the logical name.

The user privilege SYSNAM is required to place a name in the system logical name table.

/USER_MODE
Specifies, for an entry in the process logical name table, that the logical name be entered in user mode.

A user mode logical name is practical for the execution of a single image; it allows an image executing in a command procedure to redefine SYS$INPUT. User mode entries are deleted when any image executing in the process exits (that is, after any DCL command or user program that executes an image completes execution), or when a STOP command is issued.

By default, process logical name table entries are made in supervisor mode. The /USER_MODE qualifier is ignored when entries are made in the group or system logical name tables.

Examples

1.
```
$ DEFINE PROCESS_NAME LIBRA
$ RUN WAKE
```

   The DEFINE command places the logical name PROCESS_NAME in the process logical name table with an equivalence name of LIBRA. The program WAKE can translate the logical name PROCESS_NAME to perform some special action on the process named LIBRA.

2.
```
$ DEFINE TEMP:  DBA1:
$ DEASSIGN TEMP::
```

   The DEFINE command creates an equivalence name for the logical name TEMP: and places the name in the process logical name table. The DEASSIGN command deletes the logical name. Note that two colons are required on the logical name in the DEASSIGN command because the DEFINE command does not remove trailing colons from logical names, as the DEASSIGN command does.

DELETE

Deletes one or more files from a mass storage disk volume.

If you specify /ENTRY, the DELETE command deletes one or more entries from a printer or batch job queue. If you specify /SYMBOL, the DELETE command deletes a local or global symbol name assignment. The DELETE/ENTRY and DELETE/SYMBOL commands are each described under their own headings.

Format

```
DELETE   file-spec,...

   Command Qualifiers        Default

   /BEFORE[=time]
   /[NO]CONFIRM              /NOCONFIRM
   /CREATED
   /EXPIRED
   /[NO]LOG                  /NOLOG
   /MODIFIED
   /SINCE[=time]
```

Prompts

```
File:       file-spec,...
```

Command Parameters

file-spec,...

Specifies the names of one or more files to be deleted. The first file specification must contain a file name, file type, and version number; subsequent file specifications must contain a version number. You can specify any of these fields as wild cards.

To delete more than one file, separate the file specifications with commas (,) or plus signs (+).

Command Qualifiers

/BEFORE[=time]

Specifies that only the files dated earlier than a particular time be deleted. You can specify one of the following values:

absolute-time	An absolute date and time. Use the syntax rules for date and time values specified in Section 6.8.
TODAY	The absolute date and time representing the current day, month, and year, at 00:00 o'clock.
YESTERDAY	The absolute date and time representing 1 nanosecond before the most recently passed midnight.

If you specify /BEFORE and do not specify a value, the DELETE command assumes /BEFORE=TODAY.

Use the /CREATED, /EXPIRED, or /MODIFIED qualifiers to request that only files created, expired, or modified before the specified time be deleted. If none of these qualifiers is specified, the DELETE command deletes all files created and modified within the specified time.

/CONFIRM
/NOCONFIRM

Controls whether the DELETE command displays the file specification of each file before deleting and requests you to confirm whether or not the file should actually be deleted. If you specify /CONFIRM, you must respond to a prompt with a Y, followed by a carriage return, before the DELETE command will delete the file. If you enter anything else, the file is not deleted.

By default, the DELETE command does not request confirmation of files it is deleting.

/CREATED

Specifies, when /BEFORE and/or /SINCE is specified, that only files created within the defined time period be deleted.

/MODIFIED is the default if none of the qualifiers /CREATED, /MODIFIED, or /EXPIRED is specified.

/EXPIRED

Specifies, when /BEFORE and/or /SINCE is specified, that only files that reached their expiration dates within the specified time be deleted.

If any file does not have an expiration date associated with it, it is assumed to have expired at the time the DELETE command is issued. (Files can be assigned expiration dates when they are created or revised with RMS.)

/LOG
/NOLOG

Controls whether the DELETE command displays the file specification of each file that it deletes.

By default, the DELETE command does not display the names of files as it deletes them.

/MODIFIED

Specifies, when /BEFORE and/or /SINCE are specified, that only files that were modified within the defined time period be deleted. A file's revision date is updated whenever the file is accessed and updated.

/SINCE[=time]
Specifies that only the files dated later than a particular time be deleted. You can specify one of the following values:

absolute-time	An absolute date and time. Use the syntax rules for date and time values specified in Section 6.8.
TODAY	The absolute date and time representing the current day, month, and year, at 00:00 o'clock.
YESTERDAY	The absolute date and time representing 1 nanosecond before the most recently passed midnight.

If you specify /SINCE and do not specify a value, the DELETE command assumes /SINCE=YESTERDAY.

Use the /CREATED, /EXPIRED, or /MODIFIED qualifiers to request that only files created, expired, or modified after the specified time be deleted. If none of these qualifiers is specified, the DELETE command deletes all files created and modified within the specified time.

Examples

1.
```
$ DELETE   COMMON.SUM;2
```

The DELETE command deletes the file COMMON.SUM;2 from the current default disk and directory.

2.
```
$ DELETE *.OLD;*
```

The DELETE command deletes all versions of files with file types of OLD from the default disk directory.

3.
```
$ DELETE  ALPHA.TXT;*,BETA;*,GAMMA;*
```

The DELETE command deletes all versions of the files ALPHA.TXT, BETA.TXT, and GAMMA.TXT. The command uses the file type of the first input file as a temporary default. Note, however, that version numbers (here specified as wild cards) must be included in each file specification.

4.
```
$ DELETE *.DAT;*/BEFORE=01-JUN/LOG
%DELETE-I-DELETED, DBA2:[MALCOLM]ASSIGN.DAT;1 deleted
%DELETE-I-DELETED, DBA2:[MALCOLM]BATCHAVE.DAT;1 deleted
%DELETE-I-DELETED, DBA2:[MALCOLM]CANCEL.DAT;1 deleted
%DELETE-I-DELETED, DBA2:[MALCOLM]DEFINE.DAT;1 deleted
%DELETE-I-DELETED, DBA2:[MALCOLM]EXIT.DAT;1 deleted
```

THE DELETE command deletes all versions of all files with file types of COM that were either created or updated before June 1, this year.

5.
```
$ DELETE/CONFIRM [MALCOLM.TESTFILES]*.OBJ;*/SINCE=YESTERDAY
DBA2:[MALCOLM.TESTFILES]AVERAG.OBJ;1, delete? (Y or N):Y
DBA2:[MALCOLM.TESTFILES]SCANLINE.OBJ;2, delete? (Y or N):N
DBA2:[MALCOLM.TESTFILES]WEATHER.OBJ;3, delete? (Y or N):Y
```

The DELETE command examines all versions of files with file types of OBJ in the subdirectory [MALCOLM.TESTFILES], and locates those that were created or modified since yesterday, that is, that were created today. Before deleting each file, it requests confirmation that the file should be deleted.

6.
```
$ DIRECTORY [.SUBTEST]
PIP -- no such file(s)

$ SET PROTECTION SUBTEST.DIR/PROTECTION=OWNER:D
$ DELETE SUBTEST.DIR;1
```

Before deleting the directory file SUBTEST.DIR, the DIRECTORY command is used to verify that there are no files cataloged in the directory. The SET PROTECTION command redefines the protection for the directory file so that it can be deleted; then, the DELETE command deletes it.

DELETE/ENTRY

Deletes one or more entries from a printer or batch job queue. The /ENTRY qualifier is required.

Format

```
DELETE/ENTRY=jobid,...  queue-name
```

Additional Command Qualifiers

None.

Prompts

Queue: queue-name

Command Parameters

queue-name
Specifies the name of the queue in which the job(s) exist.

Command Qualifiers

/ENTRY=jobid,...
Specifies the jobid(s) of one or more jobs to be deleted from a printer or batch job queue. If you specify more than one jobid, separate them with commas and enclose the list in parentheses.

The job(s) to be deleted must have been queued by the current process or any process in the same group as the current process.

You can delete jobs that have not yet begun processing or files that are currently being processed.

Examples

1.
```
$ PRINT/HOLD   ALPHA.TXT
  Job 110 entered on queue SYS$PRINT
  .
  .
$ DELETE/ENTRY=110  SYS$PRINT
```

The PRINT command queues a copy of the file ALPHA.TXT in a HOLD status, to defer its printing until a later time. The system displays the jobid, 110, and the name of the queue in which the file was entered. Later, the DELETE/ENTRY command requests that the entry be deleted from the queue SYS$PRINT.

2.
```
$ SUBMIT/HOLD/PARAMETERS=SCANLINE  DOFOR
  Job 203 entered on queue SYS$BATCH
$ SUBMIT/AFTER=18:00  WEATHER
  Job 210 entered on queue SYS$BATCH
    .
    .
$ DELETE/ENTRY=(203,210)    SYS$BATCH
```

The SUBMIT commands spool the command procedures DOFOR.COM and WEATHER.COM for processing as batch jobs. DOFOR.COM is queued in a HOLD status; WEATHER.COM is queued for execution after 6:00 P.M. Later, the DELETE/ENTRY command requests that both these entries be deleted from the queue SYS$BATCH.

DELETE/SYMBOL

Deletes a symbol definition from a local symbol table or from the global symbol table, or deletes all symbol definitions in a symbol table.

Format

```
DELETE/SYMBOL   symbol-name

    Command Qualifiers        Default

    /ALL
    /GLOBAL                   /LOCAL
    /LOCAL                    /LOCAL
```

Prompts

Symbol: symbol-name

Command Parameters

symbol-name
Specifies the name of the symbol to be deleted. By default, the DELETE/SYMBOL command assumes the symbol is in the local symbol table for the current command procedure.

The symbol-name parameter is required unless /ALL is specified.

Description

If you specify both of the qualifiers /GLOBAL and /LOCAL, only the last one entered is accepted. The /SYMBOL qualifier must follow the DELETE command name.

Command Qualifiers

/ALL
Specifies that all symbol names in the specified symbol table be deleted. If you do not specify either /LOCAL or /GLOBAL, all symbols defined at the current command level are deleted.

/GLOBAL
Indicates that the specified symbol name is in the global symbol table for the current process.

/LOCAL
Indicates that the symbol name is in the local symbol table for the current command level.

Examples

1. `$ DELETE/SYMBOL/ALL`

   The DELETE/SYMBOL command deletes all symbol definitions at the current command level.

2. `$ DELETE/SYMBOL/GLOBAL PDEL`

   The DELETE/SYMBOL command deletes the symbol named PDEL from the global symbol table for the process.

DEPOSIT

Replaces the contents of a specified location or locations in virtual memory.

The DEPOSIT command, together with the EXAMINE command, aids in debugging programs interactively. The DEPOSIT command is similar to the DEPOSIT command of the VAX-11 Symbolic Debugger.

You can truncate the DEPOSIT command to a single letter, D.

Format

```
DEPOSIT    location = data,...

   Command Qualifiers

   /ASCII
   /BYTE
   /DECIMAL
   /HEXADECIMAL
   /LONGWORD
   /OCTAL
   /WORD
```

Prompts

None.

Command Parameters

location

Specifies the starting virtual address of a location or series of locations whose contents are to be changed.

The specified location must be within the virtual address space of the image currently running in the process, and it must be accessible for both reading and writing for user access mode.

You can specify the location using any valid arithmetic expression. The expression can contain arithmetic or logical operators or symbol names which have been previously given values with DCL assignment statements. The DEPOSIT command automatically substitutes symbols with their current values when it evaluates the specified location.

The DEPOSIT and EXAMINE commands maintain a pointer to a current memory location. The DEPOSIT command sets this pointer to the byte following the last byte modified; you can refer to this pointer using the symbol "." in subsequent EXAMINE and DEPOSIT commands. If the DEPOSIT command cannot deposit the specified data, the pointer does not change. The EXAMINE command does not change the value of the pointer.

data,...
Defines the data to be deposited into the specified location(s). If you specify a list, separate the items with commas; the DEPOSIT command writes the data in consecutive locations, beginning with the address specified.

By default, the data is assumed to be in hexadecimal format; the DEPOSIT command converts the data to binary format before writing it into the specified location. When non-ASCII data is deposited, the DEPOSIT command automatically performs symbol substitution when it evaluates data.

Description

When the DEPOSIT command completes, it displays the virtual memory address into which data is deposited and displays the new contents of the location, as follows:

address: contents

If the address specified can be read, but not written, by the current access mode, the DEPOSIT command displays the original contents of the location. If the address specified can be neither read nor written, the DEPOSIT command displays asterisks (****) in the data field.

If you specify a list of numeric values, some, but not all, of the values may be successfully deposited before an access violation occurs. If an access violation occurs while ASCII data is being deposited, nothing is deposited.

Radix Qualifiers: The radix default for a DEPOSIT or EXAMINE command determines how the command interpreter interprets numeric literals, for example, 256. The initial default radix is hexadecimal; all numeric literals in the command line are assumed to be hexadecimal values. If a radix qualifier modifies the command, that radix becomes the default for subsequent EXAMINE and DEPOSIT commands, until another qualifier overrides it. For example:

```
$ DEPOSIT/DECIMAL 900 = 256
00000388:  256
```

The DEPOSIT command interprets both the location 900 and the value 256 as decimal. All subsequent DEPOSIT and EXAMINE commands assume that numbers you enter for addresses and data are decimal. Note that the DEPOSIT command always displays the address location in hexadecimal.

Symbol values defined by = (Assignment Statement) commands are always interpreted in the radix in which they were defined.

Note that hexadecimal values entered as deposit locations or as data to be deposited must begin with a numeric character (0 through 9). Otherwise, the command interpreter assumes that you have entered a symbol name and attempts symbol substitution.

You can use the radix operators %X, %D, or %O to override the current default when you enter the DEPOSIT command. For example:

```
$ DEPOSIT/DECIMAL %X900 = 10
```

This command deposits the decimal value 10 in the location specified as hexadecimal 900.

Length Qualifiers: The initial default length unit for the DEPOSIT command is a longword. If a list of data values is specified, the data is deposited into consecutive longwords beginning at the specified location. If a length qualifier modifies the command, that length becomes the default for subsequent EXAMINE and DEPOSIT commands, until another qualifier overrides it.

Restriction on Placement of Qualifiers: The DEPOSIT command analyzes expressions arithmetically. Therefore, qualifiers (which must be preceded by /) are interpreted correctly only when they appear immediately after the command name.

Command Qualifiers

/ASCII

Indicates that the specified data is ASCII. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

When you specify ASCII data, the command interpreter compresses multiple blanks to a single blank before writing the data in memory; to deposit an ASCII string containing multiple blanks, enclose the string in quotation marks (").

When you specify /ASCII, or when ASCII mode is the default, any literal numeric values you enter are assumed to be hexadecimal.

/BYTE

Requests that data be deposited one byte at a time.

If you specify data values that are longer than a byte, an error occurs.

/DECIMAL

Indicates that the specified data is decimal; the DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/HEXADECIMAL

Indicates that the specified data is hexadecimal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/LONGWORD

Requests that data be deposited a longword at a time.

/OCTAL

Indicates that the specified data is octal; the DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/WORD

Requests that the data be deposited a word at a time.

Examples

1.
```
$ RUN MYPROG
    .
    .
^Y
$ EXAMINE 2780
00002780:  1C50B344
$ DEPOSIT .  = 0
00002780:  00000000
$ CONTINUE
```

The RUN command executes the image MYPROG.EXE; subsequently, CTRL/Y interrupts the program. Assuming that the initial defaults of /HEXADECIMAL and /LONGWORD are in effect, the DEPOSIT command places a longword of 0s in virtual memory location 2780.

The CONTINUE command resumes execution of the image.

2.
```
$ DEPOSIT/ASCII   2C00   = FILE: NAME: TYPE:
00002C00:  FILE: NAME: TYPE:...
```

The DEPOSIT command deposits character data at hexadecimal location 2C00 and displays the contents of the location after modifying it. Since the current default length is a longword, the response from the DEPOSIT command displays full longwords. Trailing dots indicate that the remainder of the last longword of data contains information that was not modified by the DEPOSIT command.

3.
```
$ EXAMINE 9C0          !  Look at Hex location 9C0
000009C0:  8C037DB3
$ DEPOSIT . = 0        !  Deposit longword of 0
000009C0:  00000000
$ DEPOSIT/BYTE . = 1 !  Put 1 byte at next location
000009C4:  01
$ DEPOSIT .+2 = 55    !  Deposit 55 next
000009C7:  55
$ DEPOSIT/LONG .  = 0C, 0D, 0E !  Deposit longwords
000009C8:  0000000C 0000000D 0000000E
```

The sequence of DEPOSIT commands in the above example illustrates how the DEPOSIT command changes the current position pointer. Note that after you specify /BYTE, all data is deposited and displayed in bytes, until the /LONGWORD qualifier restores the system default.

4.
```
$ BASE = %X200          !  Define a base address
$ LIST = BASE+%X40     !  Define offset from base
$ DEPOSIT/DECIMAL LIST = 1,22,333,4444
00000240:  00000001 00000022 00000333 00004444
$ EXAMINE/HEX LIST:LIST + 0C !  Display results in hex
00000240:  00000001 00000016 0000014D 0000115C
```

The assignment statements define a base address in hexadecimal and a label at a hexadecimal offset from the base address. The DEPOSIT command reads the list of values and deposits each value into a longword, beginning at the specified location. The EXAMINE command requests a hexadecimal display of these values.

DIFFERENCES

Compares the contents of two disk files and creates a listing of the records that do not match.

Format

```
DIFFERENCES   input-file-spec  [compare-file-spec]

   Command Qualifiers                          Default

   /COMMENT_DELIMITERS[=characters,...]
   /IGNORE=characters,...
   /LINE_WIDTH=n
   /MATCH=size                                 /MATCH=3
   /MAXIMUM_DIFFERENCES=n
   /MERGED[=n]                                 /MERGED=1
   /MODE=radix,...                             /MODE=ASCII
   /OUTPUT=file-spec                           /OUTPUT=SYS$OUTPUT
   /SEPARATED                                  /NOSEPARATED
   /SLP
   /WINDOW=size                                /WINDOW=15


   File Qualifiers

   /CHANGE_BAR[=[c][,[NO]NUMBER]]
```

Prompts

File 1: input-file-spec

File 2: compare-file-spec

Command Parameters

input-file-spec
Specifies the name of the primary input file to be compared.

The file specification must include a file name and file type. No wild cards are allowed in the file specification.

compare-file-spec
Specifies the name of the secondary input file to be compared. Any nonspecified fields default to the corresponding field of the primary input file specification.

If you do not specify a secondary input file, the DIFFERENCES command uses the next lowest version of the primary input file.

Description

Use the DIFFERENCES command to find out whether two files are identical and, if not, how they differ. DIFFERENCES compares the two files specified, on a record-by-record basis, and produces an output file that lists the differences, if any.

The qualifiers for the DIFFERENCES command can be categorized according to their functions, as follows.

- Qualifiers that request DIFFERENCES to ignore data in each record are:

  /COMMENT_DELIMITERS
  /IGNORE

  These qualifiers allow you to define characters that denote comments and characters to ignore while comparing files -- for example, extra blank lines, or tabs within lines.

  By default, DIFFERENCES compares every character in each record.

- Qualifiers that control the format of the information produced in the list of differences are:

  /CHANGE_BAR
  /LINE_WIDTH
  /MERGE
  /MODE
  /SEPARATED
  /SLP

  By default, DIFFERENCES merges differences it finds in the files being compared, and lists each record in the input file that has no match in the output file, followed by the first record that it finds that does have a match.

  You can specify combinations of qualifiers to request an output listing from DIFFERENCES that includes the comparison in more than one format. However, SLP output is incompatible with all other types of output.

- Qualifiers that control the extent of the comparison are:

  /MATCH
  /MAXIMUM_DIFFERENCES
  /WINDOW

  By default, DIFFERENCES reads every record in the primary input file, and looks for a matching record in the secondary input file. It terminates each search if it does not find a match within 15 records. Records are considered matched only if three sequential records are found in each file that are identical.

DIFFERENCES output is written by default to the current output device, that is, to SYS$OUTPUT. Use the /OUTPUT qualifier to request DIFFERENCES to write the output to an alternate file or device.

Command Qualifiers

/COMMENT_DELIMITERS[=characters,...]

Requests that lines that are comments not be included in the comparison. If a specified comment character or characters appears as the first character in an input record, DIFFERENCES ignores the record in the comparison.

You can specify up to four comment characters. If you specify more than one, separate the characters with commas and enclose the list in parentheses.

You can specify characters either by using the character itself or by using one of the following keywords:

Keyword	Character
COLON	Colon (:)
COMMA	Comma (,)
EXCLAMATION	Exclamation point (!)
FORM	Form feed
LEFT	Left bracket ([)
RIGHT	Right bracket (])
SEMI	Semi-colon (;)
SLASH	Slash (/)
SPACE	Space
TAB	Horizontal tab

If you do not include a comment character, DIFFERENCES assumes the following default comment characters for the associated file types:

File Type	Default Comment Character
CBL	* and ;
CMD	! and ;
COM	!
FOR	! and C and D
All others	;

If the comment character (either explicitly or by default) is either an exclamation point (!) or semicolon (;), DIFFERENCES also ignores any comments on the right-hand side of a statement. If you also specify /IGNORE=TRAILING_BLANKS, DIFFERENCES ignores multiple blank spaces or tabs immediately preceding the comment character as well.

/IGNORE=characters,...

Specifies one or more special characters to be ignored during the comparision. You can request DIFFERENCES to ignore the following:

BLANK_LINES — Blank lines between data lines

COMMENTS — Lines beginning with a comment character (use the /COMMENT_DELIMITERS qualifier to designate one or more non-default comment characters)

FORM_FEEDS — Form feed characters (DIFFERENCES removes form feed characters before comparing records)

TRAILING_BLANKS — Extra blank characters at the end of a line of text (DIFFERENCES strips trailing blanks and tabs before comparing records)

SPACING — Extra blank spaces or tabs within lines of text (DIFFERENCES compresses multiple blanks or tabs to a single blank before comparing records)

By default, the DIFFERENCES command compares every character in each file and reports all differences.

/LINE_WIDTH=n
Specifies the width of lines in the output listing.

The minimum value for the line width, n, is 30.

By default, output is 132 characters wide, unless output is directed to the terminal. In this case, the output line width is controlled by the terminal line width.

/MATCH=size
Controls the number of records to be included in a match. The size value can be specified as a decimal number between 2 and 20.

By default, after DIFFERENCES finds unmatched records, it assumes that the files match after it finds 3 sequential records that match.

Specify a match size to override the default value of 3.

/MAXIMUM_DIFFERENCES=n
Specifies that the command is to terminate after n unmatching records have been found.

If you specify /MAXIMUM_DIFFERENCES, DIFFERENCES terminates after locating n unmatched records. The output file lists differences only on records compared until the maximum has been reached.

By default, DIFFERENCES compares every record in the specified input file.

/MERGED[=n]
Requests that the output file contain a merged list of differences. The value n is a decimal number from 1 to 10 indicating the number of matched records to list after each list of unmatched records.

By default, DIFFERENCES produces a merged listing, with one matched line listed after each set of unmatched records.

Use /MERGED to override the default value of n, or to include a merged listing with other types of output.

/MODE=radix,...
Specifies the format of the output listing. You can request that the output be formatted in one or more radix modes by specifying the following keywords:

ASCII
HEXADECIMAL
OCTAL

You can truncate any of these keywords to 1 or more characters.

By default, DIFFERENCES writes the output file in ASCII. If you specify more than one code, the output listing contains the file comparison in each output mode.

If you specify /SLP, the /MODE qualifier is ignored.

/OUTPUT=file-spec
Defines an output file to receive the output difference list. By default, the output is written to the current default output device (SYS$OUTPUT). When you specify /OUTPUT, you can control the defaults applied to the output file specification, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers." The output file type always defaults to DIF.

/SEPARATED
/NOSEPARATED

Controls whether DIFFERENCES creates a listing that contains a sequential list of unmatched records. Unmatched records in the primary input file are listed first, followed by unmatched records in the compare file.

By default, DIFFERENCES creates only a merged list of differences.

/SLP

Requests DIFFERENCES to produce an output file suitable for input to the SLP editor. If you specify /SLP you cannot specify any of the following output file qualifiers: /MERGED, /SEPARATED, /CHANGE_BAR.

Use the output file produced by the /SLP qualifier as input to SLP to update the primary input file specified, that is, to make the first input file match the second input file.

When you specify /SLP and you do not specify /OUTPUT, DIFFERENCES writes the output file to a file with the same file name as the primary input file and a filetype of DIF.

/WINDOW=size

Controls the number of records to search before listing a record as unmatched.

The window size is a decimal number with a minimum value of 5 and no maximum.

By default, the window size is 15, that is, DIFFERENCES searches 15 records in the compare file before listing a record as unmatched.

Specify a window size to control the number of records to search before listing the record as unmatched and continuing with the next record in the input file.

File Qualifiers

/CHANGE_BAR[=[c][,[NO]NUMBER]]

Requests that the output contain a listing of the associated file(s) with a change bar character next to the lines in the file that do not match.

The change bar character, c, specifies a one-character code that will appear in the left margin next to records that do not have a match.

By default, an exclamation point (!) is used as the change bar character.

You can also control whether the listing includes line numbers. You can specify:

NUMBER	Print line numbers
NONUMBER	Do not print line numbers

If not specified, NUMBER is the default; that is line numbers are printed in the listing.

To specify both a change bar character and to request no numbers, separate the options with a comma and enclose them in parentheses, for example, /CHANGE_BAR=(*,NONUMBER).

Examples

Figure 1 shows the output from the DIFFERENCES command examples described below. Unless otherwise noted, the output is displayed on the current SYS$OUTPUT device.

1.
```
$ DIFFERENCES COPYDOC.COM
```

   The DIFFERENCES command compares the contents of the two most recent versions of the file COPYDOC.COM in the current default directory. DIFFERENCES compares every character in every record and displays the results in the terminal.

2.
```
$ DIFFERENCES/IGNORE=(COMMENTS,SPACING) COPYDOC.COM
```

   The DIFFERENCES command compares the same files, but ignores all comment lines (that is, all lines beginning with exclamation points), and ignores multiple blanks or tabs in input lines.

3.
```
$ DIFFERENCE /OUTPUT -
$_ COPYDOC.COM/SEPARATED/CHANGE_BAR=NONUMBER
DIF OUTPUT IN FILE SY:COPYDOC.DIF;1
```

   The DIFFERENCES command compares the same files, but requests two listings: one that lists the differences in each file separately, rather than merging them; and one that lists the first input file with change bar characters next to the lines that do not have a match in the second input file. The /CHANGE_BAR qualifier is a file qualifier; thus, only the file COPYDOC.COM;2 (the primary input file, by default) is listed with change bars indicating the lines that do not have an exact match in the file COPYDOC.COM;1.

   Both types of output are written to the requested file, COPYDOC.DIF.

4.
```
$ DIFFERENCES COPYDOC.COM [MALCOLM.TESTFILES]
```

   The DIFFERENCES command compares the highest existing versions of the file COPYDOC.COM in the current default directory with the copy in the directory MALCOLM.TESTFILES.

Output for Example 1

```
******************************************************************************
************************* FILE COMPARE UTILITY ***************************
************************** DIF -- VERSION 1.0 ****************************
******************************************************************************

******************************************************************************
********************** MERGED LIST OF DIFFERENCES ************************
******************************************************************************

FILE SY:COPYDOC.COM;3
    1  $ ! Command procedure to copy all files with file type of TXT
    2  $ ! into a master file named MASTER.DOC and to print 20 copies
    3  $ ! of MASTER.DOC
****************************
FILE SY:COPYDOC.COM;2
    1  $ ! Command Procedure to copy all files with file type of TXT
    2  $ ! into a master file named MASTER.DOC and to print 10 copies
    3  $ ! of MASTER.DOC
**************************************************
**************************************************
FILE SY:COPYDOC.COM;3
    6  $ DELETE MASTER.DOC;*
    7  $ PURGE *.TXT
    8  $ COPY *.TXT    MASTER.DOC
    9  $ PRINT/COPIES=20/LOWER MASTER.DOC
****************************
FILE SY:COPYDOC.COM;2
    6  $ PURGE *.TXT
    7  $ COPY *.TXT MASTER.DOC
    8  $ PRINT/COPIES=10/LOWER MASTER.DOC
```

Output for Example 2

```
******************************************************************************
************************* FILE COMPARE UTILITY ***************************
************************** DIF -- VERSION 1.0 ****************************
******************************************************************************

******************************************************************************
********************** MERGED LIST OF DIFFERENCES ************************
******************************************************************************

FILE SY:COPYDOC.COM;3
    6  $ DELETE MASTER.DOC;*
    7  $ PURGE *.TXT
    8  $ COPY *.TXT    MASTER.DOC
    9  $ PRINT/COPIES=20/LOWER MASTER.DOC
****************************
FILE SY:COPYDOC.COM;2
    6  $ PURGE *.TXT
    7  $ COPY *.TXT MASTER.DOC
    8  $ PRINT/COPIES=10/LOWER MASTER.DOC
```

Figure 1 Sample Output of DIFFERENCES Command

Output for Example 3

```
*****************************************************************************
************************* FILE COMPARE UTILITY ******************************
************************** DIF -- VERSION 1.0 *******************************
*****************************************************************************

*****************************************************************************
********************** CHANGE-BAR OUTPUT FOR FILE ***************************
************************** SY:COPYDOC.COM;3 **********************************
*****************************************************************************

!       $ ! Command procedure to copy all files with file type of TXT
!       $ ! into a master file named MASTER.DOC and to print 20 copies
        $ ! of MASTER.DOC
        $ !
        $ !
!       $ DELETE MASTER.DOC;*
!       $ PURGE *.TXT
!       $ COPY *.TXT     MASTER.DOC
!       $ PRINT/COPIES=20/LOWER MASTER.DOC

*****************************************************************************
************************** LISTED DIFFERENCES *******************************
*****************************************************************************

                  RECORDS FROM FILE SY:COPYDOC.COM;3
                WITHOUT A MATCH IN FILE SY:COPYDOC.COM;2

    1   $ ! Command procedure to copy all files with file type of TXT
    2   $ ! into a master file named MASTER.DOC and to print 20 copies
    6   $ DELETE MASTER.DOC;*
    7   $ PURGE *.TXT
    8   $ COPY *.TXT     MASTER.DOC
    9   $ PRINT/COPIES=20/LOWER MASTER.DOC

                  RECORDS FROM FILE SY:COPYDOC.COM;2
                WITHOUT A MATCH IN FILE SY:COPYDOC.COM;3

    1   $ ! Command Procedure to copy all files with file type of TXT
    2   $ ! into a master file named MASTER.DOC and to print 10 copies
    6   $ PURGE *.TXT
    7   $ COPY *.TXT MASTER.DOC
    8   $ PRINT/COPIES=10/LOWER MASTER.DOC
```

Output for Example 4

```
*****************************************************************************
************************* FILE COMPARE UTILITY ******************************
************************** DIF -- VERSION 1.0 *******************************
*****************************************************************************

*****************************************************************************
********************** MERGED LIST OF DIFFERENCES ***************************
*****************************************************************************

                NONE
```

Figure 1 (Cont.) Sample Output of DIFFERENCES Command

DIRECTORY

Provides a list of files or information about a file or group of files.

Format

```
DIRECTORY   [file-spec,...]

   Command Qualifiers          Default

   /BRIEF
   /FULL
   /OUTPUT=file-spec           /OUTPUT=SYS$OUTPUT
   /PRINTER
```

Prompts

None.

Command Parameters

file-spec,...
Specifies one or more files to be listed. The syntax of a file specification determines what file(s) are listed, as follows:

- If you do not enter a file specification, the DIRECTORY command lists all files in your current default directory.
- If you specify only a device name, the DIRECTORY command uses your default directory name.
- Whenever the file specification does not include a file name and file type, all versions of all files in the specified directory are listed.
- If a file specification contains a file name and/or file type and no version number, the DIRECTORY command lists the highest existing version.
- If a file specification contains only a file name, the DIRECTORY command assumes a null file type.

If you specify more than one file, separate the file specifications with either commas (,) or plus signs (+). You can use wild cards in place of the directory, file name, file type, or version fields of a file specification to list all files that satisfy the components you specify.

Command Qualifiers

/BRIEF
Lists only the file name, type, and version of each file to be listed, as shown in Figure 2.

/FULL
Lists all information about each file to be listed, as shown in Figure 2.

/OUTPUT=file-spec
Requests that the DIRECTORY command output be written to the file specified rather than to the current SYS$OUTPUT device.

/PRINTER
Queues a copy of the command output to the system printer. The output file is identified as DIRECT.LOG. If you specify /PRINTER, you cannot specify /OUTPUT.

Examples

1. `$ DIRECTORY`

   The DIRECTORY command lists all files in the current default disk and directory.

2. `$ DIRECTORY LOGIN.COM;*`

   The DIRECTORY command lists all versions of the file LOGIN.COM.

3. `$ DIRECTORY ALPHA.TXT,BETA,GAMMA`

   The DIRECTORY command lists the most recent versions of the files ALPHA.TXT, BETA.TXT, and GAMMA.TXT in the current default directory.

4. `$ DIRECTORY/FULL/PRINTER`

   The DIRECTORY command prints a full listing of all files in the current default directory.

5. `$ DIRECTORY DBA2:[*]RMSLIB.OLB`

   The DIRECTORY command lists the highest versions of all files named RMSLIB.OLB in all first level (user file) directories on the disk DBA2.

6. `$ DIRECTORY [.TESTFILES]`

   Lists all files in the subdirectory TESTFILES. The subdirectory file, TESTFILES.DIR, is cataloged in the current default directory.

The following notes are keyed to the sample DIRECTORY command listings in Figure 2.

(1) Disk and directory name

(2) Date and time the DIRECTORY command was issued

(3) File name, type, and version number of each file

(4) File identification number in the format:

    (file-number,file-sequence-number)

(5) Number of blocks occupied by the file and the number of blocks allocated for the file

6 File code. The code is one of the following:

Code	Meaning
(null)	- file is noncontiguous
C	- file is contiguous
L	- file is locked

7 Date and time the file was created or last modified

8 User identification code of the file's owner, and the protection code associated with the file, in the format:

[group,member][system,owner,group,world]

9 Summary of information, in the format:

TOTAL OF in-use./allocated BLOCKS IN XXXXX. FILES

10 Date and time that this version of the file was last revised, and the revision number

```
$ DIRECTORY AVERAGE.*

DIRECTORY DBA2:[MALCOLM] (1)
20-JUN-78 10:11 (2)

AVERAGE.COM;1 (3)     1. (5)     (6) 25-JAN-78 15:39 (7)
AVERAGE.DAT;1         1.         22-MAR-78 09:27
AVERAGE.EXE;9         45.    C   19-APR-78 09:51
AVERAGE.FOR;2         2.         19-APR-78 09:50
AVERAGE.LIS;2         5.         12-JUN-78 16:27
AVERAGE.OBJ;9         2.         12-JUN-78 16:27
AVERAGE.OUT;1         1.         19-APR-78 10:28

TOTAL OF 57./100. BLOCKS IN 7. FILES (9)


$ DIRECTORY AVERAGE.*/BRIEF

DIRECTORY DBA2:[MALCOLM] (1)

AVERAGE.COM;1 (3)
AVERAGE.DAT;1
AVERAGE.EXE;9
AVERAGE.FOR;2
AVERAGE.LIS;2
AVERAGE.OBJ;9
AVERAGE.OUT;1


$ DIRECTORY AVERAGE.*/FULL

DIRECTORY DBA2:[MALCOLM] (1)
20-JUN-78 10:11 (2)

AVERAGE.COM;1 (3)     (6334,120) (4)   1./5. (5)      (6) 25-JAN-78 15:39 (7)
(8)[122,1]  (RWED,RWED,RWED,R)
AVERAGE.DAT;1         (13123,34)       1./5.              22-MAR-78 09:27
[122,1]  (RWED,RWED,RWED,R)    19-APR-78 10:28(6.) (10)
AVERAGE.EXE;9         (22776,24)       45./70.        C   19-APR-78 09:51
[122,1]  (RWED,RWED,RWED,R)
AVERAGE.FOR;2         (22751,31)       2./5.              19-APR-78 09:50
[122,1]  (RWED,RWED,RWED,R)
AVERAGE.LIS;2         (11323,56)       5./5.              12-JUN-78 16:27
[122,1]  (RWED,RWED,RWED,R)
AVERAGE.OBJ;9         (13051,71)       2./5.              12-JUN-78 16:27
[122,1]  (RWED,RWED,RWED,R)
AVERAGE.OUT;1         (7026,41)        1./5.              19-APR-78 10:28
[122,1]  (RWED,RWED,RWED,R)

TOTAL OF 57./100. BLOCKS IN 7. FILES (9)
```

Figure 2 Sample Output of DIRECTORY Command

DISMOUNT

Releases a volume previously accessed with a MOUNT command.

Format

```
DISMOUNT   device-name[:]

   Command Qualifiers        Default

   /[NO]UNLOAD               /UNLOAD
```

Prompts

Device: device-name

Command Parameters

device-name
: Specifies the name of the device to be dismounted. You can specify a physical device name or a logical name assigned to a physical device name. If you omit a controller designation and/or a unit number, they default to controller A, and unit 0, respectively.

Description

If the volume was mounted /SHARE, it is not actually dismounted until all users who mounted it dismount it. However, the DISMOUNT command deassigns the logical name associated with the device.

If the device was allocated with an ALLOCATE command, it remains allocated after the volume is dismounted with the DISMOUNT command. If the device was implicitly allocated by the MOUNT command, the DISMOUNT command deallocates it.

If the volume was mounted /GROUP or /SYSTEM, it is dismounted even if other users are currently accessing it. The GRPNAM and SYSNAM user privileges are required to dismount group and system volumes, respectively.

Note that the dismounting of a volume is done by the file system and is not completed until all open files on the volume have been closed. Thus, a substantial amount of time can pass between the time you issue the DISMOUNT command and the completion of the dismount. Always wait for the drive to unload before you remove the volume (you can verify that the dismount has completed by issuing the SHOW DEVICES command).

Command Qualifiers

/UNLOAD
/NOUNLOAD

Controls whether the DISMOUNT command unloads the physical device on which the volume is mounted and makes the device not ready.

By default, the DISMOUNT command unloads the device; use the /NOUNLOAD qualifier to keep the device and volume in a ready state.

Examples

1.
```
$ MOUNT MT:  PAYVOL TAPE
    .
    .
$ DISMOUNT TAPE
```

The MOUNT command mounts the tape whose volume identification is PAYVOL on the device MTA0: and assigns the logical name TAPE to the device. By default, the volume is not shareable. The DISMOUNT command releases access to the volume, deallocates the device, and deletes the logical name TAPE.

2.
```
$ MOUNT/SHARE DBA3:  DOC_FILES
    .
    .
$ DISMOUNT DBA3:
```

The MOUNT command mounts the volume labeled DOC_FILES on the device DBA3. Other users can issue MOUNT commands to access the device. The DISMOUNT command shown in this example deaccesses the device for the process issuing the command. If other users still have access to the volume, the volume remains mounted.

3.
```
$ DIRECTORY DMA2:[*]
PIP -- no such files
$ DISMOUNT/NOUNLOAD DMA2:
$ INITIALIZE DMA2:  BACK_UP
```

The DIRECTORY command ensures that no files remain on the RK06/RK07 volume mounted on the device DMA2. The DISMOUNT dismounts the volume; the /NOUNLOAD qualifier requests that the volume remain in a ready state. Then, the INITIALIZE command reinitializes the volume.

DUMP

Displays or prints the contents of a file or volume in ASCII, decimal, hexadecimal, or octal data format.

Format

```
DUMP   file-spec

   Command Qualifiers          Default

   /ASCII                      /HEXADECIMAL
   /BLOCKS=(START:n,END:m)
   /BYTE                       /WORD
   /DECIMAL                    /HEXADECIMAL
   /HEADER
   /HEXADECIMAL                /HEXADECIMAL
   /LONGWORD                   /WORD
   /NUMBER[=n]
   /OCTAL                      /HEXADECIMAL
   /OUTPUT[=file-spec]         /OUTPUT=SYS$OUTPUT
   /PRINTER
   /RECORDS
   /WORD                       /WORD
```

Prompts

File: file-spec

Command Parameters

file-spec
Specifies the file or volume whose contents are to be displayed.

Description

By default, the DUMP command formats its output in hexadecimal words. You can specify the precise format of the dump by including a radix qualifier (/ASCII, /OCTAL, /DECIMAL, or /HEXADECIMAL) and/or a length qualifier (/BYTE, /WORD, or /LONGWORD). The valid combinations of these qualifiers are:

```
/BYTE/HEXADECIMAL
/BYTE/OCTAL
/WORD/DECIMAL
/WORD/HEXADECIMAL
/WORD/OCTAL
/LONGWORD/HEXADECIMAL
```

All other combinations are invalid.

Dumping Files: When you dump files, you can request that the entire file be dumped, or you can specify the /BLOCKS qualifier to indicate a range of virtually contiguous blocks in the file to be dumped.

The volume that contains the file must be mounted.

Dumping Volumes: When you dump volumes, the /BLOCKS qualifier is required; use it to specify a range of logically contiguous blocks to be dumped. (On a disk volume, blocks are 512 bytes long. On a tape volume, each record is an individual block; the maximum block length that DUMP can handle is 2048 bytes.)

The volume to be dumped must be allocated and mounted with the /FOREIGN qualifier of the MOUNT command. You must have the user privilege LOG_IO to dump the contents of a volume.

Command Qualifiers

/ASCII
Requests that the dump be interpreted as ASCII data. When you specify /ASCII, the DUMP command prints control characters with a circumflex or up-arrow (↑) preceding them, and it prints lowercase letters with a percent sign (%) preceding their uppercase equivalents.

If you specify /ASCII, any other radix and length qualifiers are invalid.

/BLOCKS=(START:n,END:m)
Specifies a range of blocks to be dumped, where n is the starting logical block number and m is the ending block number.

By default, the DUMP command prints the entire contents of a file. The /BLOCKS qualifier is required when you are dumping the contents of a volume.

/BYTE
Requests that the output file be formatted in bytes.

If you specify /BYTE, you cannot specify /DECIMAL.

/DECIMAL
Requests that the dump be printed in decimal format.

If you specify /DECIMAL, you cannot specify /LONGWORD or /BYTE.

/HEADER
Requests that the dump include the file header. To dump only the file header, specify /BLOCKS=(END:0).

/HEXADECIMAL
Requests that the dump be printed in hexadecimal format.

/LONGWORD
Requests that the dump be formatted in longwords.

If you specify /LONGWORD, you cannot specify /OCTAL, /DECIMAL, or /ASCII.

/NUMBER[=n]
Controls line numbers assigned to records as they are dumped. If you specify /NUMBER, records in each block are numbered beginning with 0. If you specify a value for n, that number is assigned to the first line in the file.

By default, DUMP numbers all lines in the file consecutively, and does not number lines according to the blocks they are in.

/OCTAL
Requests that the dump be printed in octal format.

If you specify /OCTAL, you cannot specify /ASCII or /LONGWORD.

/OUTPUT[=file-spec]
Requests that the output listing from the DUMP command be written to the specified file or device. By default, the DUMP command displays the output on the current SYS$OUTPUT device. If you specify /OUTPUT and do not include a file specification, DUMP writes the output to a file with the same file name as the input file and a file type of DMP.

/PRINTER
Requests that output be queued to the system printer. By default, DUMP writes to the current SYS$OUTPUT device. If you specify /PRINTER, the DUMP command names the print job FILDMP.DMP. If you specify /PRINTER, you cannot specify /OUTPUT.

/RECORDS
Requests that the dump be printed a record at a time, rather than a block at a time.

/WORD
Requests that the dump be formatted in words.

If you specify /WORD, you cannot specify /ASCII.

Examples

1.
```
$ DUMP ORION.EXE

Dump of DB1:[122001.SSTEST]ORION.EXE;17 - File ID 1637,5,0
                Virtual block 0,000001 - size 512. bytes


3130 3230 0000 0000 0044 0038 0028 007C         0000
0000 0000 0000 0000 0000 0000 0000 0101         0010

          .         .         .              .
          .         .         .              .
```

The DUMP command displays the contents of the image ORION.EXE in hexadecimal format, beginning with the first block in the file.

2.
```
$ DUMP/ASCII/RECORDS  ALPHA.TXT

Dump of DB1:[122001.CLUG]ALPHA.TXT;1 - File ID 5767,60,0
                Record number 00. - size 5. bytes


000000   %A %A %A %A %A ^@ ^@ ^@ ^@ ^@ ^@ ^@ ^@ ^@ ^@ ^@


                Record number 01. - size 5. bytes


000000   %B %B %B %B %B ^@ ^@ ^@ ^@ ^@ ^@ ^@ ^@ ^@ ^@ ^@
   .             .         .          .
   .             .         .          .
```

The DUMP command displays the contents of the file ALPHA.TXT, a record at a time.

EDIT

Invokes one of the following VAX/VMS editors:

- SOS
- SLP

The SOS and SLP editors are described in detail in the VAX-11 Text Editing Reference Manual.

Format

```
EDIT/editor   file-spec
```

Editors	Default
/SLP	
/SOS	/SOS
Command Qualifiers	
/OUTPUT=file-spec	
SLP Qualifiers	
/[NO]AUDIT_TRAIL[=options,...]	/AUDIT_TRAIL=(POSITION:80 - SIZE:8)
/[NO]LIST[=file-spec]	/NOLIST
/[NO]TAB	/NOTAB
/[NO]TRUNCATE[=position]	/NOTRUNCATE
SOS Qualifiers	
/[NO]DECIDE	/NODECIDE
/[NO]EXACT	/NOEXACT
/[NO]EXPERT	/NOEXPERT
/INCREMENT=n	/INCREMENT=100
/ISAVE=n	/ISAVE=0
/[NO]LINE	/LINE
/[NO]LOWER	/LOWER
/PLINES=n	/PLINES=16
/[NO]READ_ONLY	/NOREAD_ONLY
/SAVE=n	/SAVE=0
/START=n	/START=100
/STEP=n	/STEP=100

Prompts

File: file-spec

Command Parameters

file-spec
Specifies the file to be created or edited using the specified editor. Neither editor provides any default file types; if you do not include a file type, it is null.

The file must be a disk file on a Files-11 formatted volume.

Command Qualifiers

/OUTPUT=file-spec

Defines the file specification of the output file created during the editing session, if any. If you do not specify /OUTPUT, the output file has the same file name and type as the input file, and a version number one higher than the highest existing version of the file.

/SLP

Invokes the source language input program (SLP) to update a file in batch mode.

The EDIT/SLP command line has the format:

```
$ EDIT/SLP  file-spec
```

where file-spec is the name of the file to be edited. Note that SLP does not prompt for input; after you issue the EDIT/SLP command, you can begin typing input for SLP.

/SOS

Invokes the default system editor, SOS. The command line has the format:

```
$ EDIT  file-spec
```

where file-spec is the name of the file to be edited. The file specification cannot contain wild cards. If the file does not currently exist, it is created.

SLP Qualifiers

You can specify the following qualifiers when you use the EDIT/SLP command to invoke SLP. These qualifiers can be overridden in the SLP input file. For complete details on any of these qualifiers, see the VAX-11 Text Editing Reference Manual.

/AUDIT_TRAIL[=options,...]
/NOAUDIT,TRAIL

Controls whether records in the output file from SLP contain an audit trail, and optionally defines the location of the audit trail. You can specify one or both of the following options:

POSITION:n — Define the starting character position of the audit trail; by default, the audit trail is placed in column 80. If you specify this option, SLP rounds the value, n, to the next highest tab stop.

SIZE:n — Define the number of characters in the audit trail; by default, the audit trail is 8 characters.

If you specify /NOAUDIT_TRAIL, the output file does not contain a record of changes.

/LIST[=file-spec]
/NOLIST

Controls whether SLP creates a line-numbered listing of a file. Use /LIST when you want a listing of lines in sequential order. If you do not specify a file specification, SLP uses the same file name as the input file and a file type of LST. If you enter a file specification that does not include a file type, SLP uses the default file type of LST.

/TAB
/NOTAB
Controls whether SLP places blank character spaces or tabs at the end of each record containing an audit trail. /NOTAB is the default; SLP inserts spaces at the end of each record that contains an audit trail.

/TRUNCATE[=position]
/NOTRUNCATE
Requests SLP to truncate each record in the input file at a specified column when it creates the output file. This qualifier allows you to delete an audit trail from a file previously updated with SLP.

If you do not specify a position with the /TRUNCATE qualifier, SLP truncates input records at the beginning position of the audit trail.

SOS Qualifiers

You can specify the following qualifiers when you use the EDIT command to invoke SOS.

The settings defined by some of these qualifiers can be overridden during the SOS session. For complete details on these qualifiers, see the VAX-11 Text Editing Reference Manual.

/DECIDE
/NODECIDE
Controls whether SOS automatically enters Decide mode following each Substitute command.

/EXACT
/NOEXACT
Controls whether SOS matches character strings in Find and Substitute commands exactly or treats uppercase and lowercase letters as equivalent.

/EXPERT
/NOEXPERT
Controls whether SOS displays the long form of error messages, requests confirmation of deletions, or displays various informational messages during the terminal session.

/INCREMENT=n
Specifies the line number increment you want SOS to use as the default when you insert new lines in the file.

/ISAVE=n
Requests SOS to issue a Save World command automatically after every n new lines of text that you input.

/LINE
/NOLINE
Indicates whether SOS should use the existing line numbers when you edit a file, or should renumber the lines when it opens the file for editing. By default, SOS uses the current line numbers in the file.

/LOWER
/NOLOWER
Indicates whether SOS should accept all lowercase letters as they are entered or should translate all lowercase letters to uppercase. By default, SOS accepts lowercase letters. The /LOWER qualifier has no effect on data that already exists in a file.

/PLINES=n
Specifies the number of lines that SOS prints each time you issue the Print command.

/READ_ONLY
/NOREAD_ONLY
Controls whether SOS opens the input file for reading and writing or only for reading. By default, SOS opens files for reading and writing.

/SAVE=n
Requests SOS to automatically issue a Save World command after every n SOS commands that change text.

/START=n
Specifies the line number you want to assign to the first line in the file and to each new page in the file. This value also controls the line number increment SOS uses when you issue the reNumber command.

/STEP=n
Specifies the line number increment for SOS to use when it assigns line numbers to existing files that do not have line numbers.

Examples

1. `$ EDIT/OUTPUT=TEST.FOR  ACCOUNT.FOR/PLINES=10`

   The EDIT command invokes the default editor, SOS, to edit the file ACCOUNT.FOR. The /PLINES qualifier sets the default number of lines to print with each Print command during the editing session. When the edit session is terminated, the changes are written into the file TEST.FOR.

2.
```
$ EDIT/SLP   AVERAGE.FOR
    .
    .
    .
/
$
```

   The command procedure illustrated uses the EDIT/SLP command; all input lines for the SLP editor following the command in the input stream, terminated by the slash character (/).

EOD

Signals the end of a data stream when a command or program is reading data from an input device other than an interactive terminal. This command is required only if the DECK command preceded input data in the command stream, or if multiple input files are contained in the command stream without intervening commands. The program or command reading the data receives an end-of-file condition when the EOD command is read.

The EOD command must be preceded by a $; the $ must be in the first character position (column 1) of the input record.

Format

```
EOD

  Command Qualifiers

  None.
```

Prompts

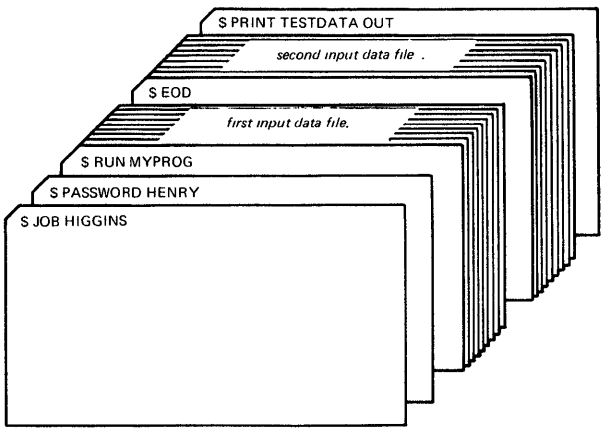
None.

Command Parameters

None.

Example

1.



The program MYPROG requires two input files; these are read from the logical device SYS$INPUT. The EOD command signals the end of the first data file and the beginning of the second. The next line that begins with a dollar sign (a PRINT command in this example) signals the end of the second data file.

For additional examples, see the discussion of the DECK command.

EOJ

Marks the end of a batch job submitted through the system card reader. The first non-blank character in the command line must be a dollar sign ($). The EOJ command performs the same functions as the LOGOUT command.

Format

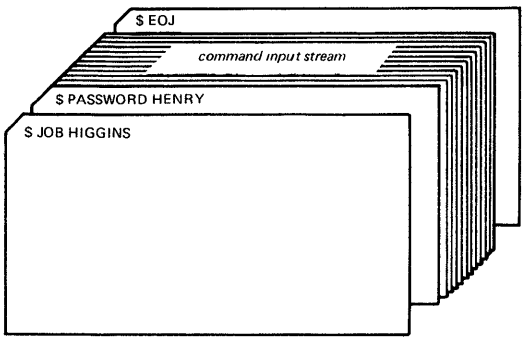
```
EOJ

  Command Qualifiers

  None.
```

Prompts

None.

Example

1.



The JOB and PASSWORD commands mark the beginning of a batch job submitted through the card reader; the EOJ command marks the end of the job.

EXAMINE

Displays the contents of virtual memory at the terminal.

You can truncate the EXAMINE command to a single letter, E.

Format

```
EXAMINE   location[:location]

  Command Qualifiers

  /ASCII
  /BYTE
  /DECIMAL
  /HEXADECIMAL
  /LONGWORD
  /OCTAL
  /WORD
```

Prompts

None.

Command Parameters

location[:location]

Specifies a virtual address or a range of virtual addresses whose contents you want to examine. If you specify a range of addresses, separate them with a colon (:); the second address must be larger than the first.

You can specify locations using any valid arithmetic expression that contains arithmetic or logical operators or symbol names which have been previously given values with DCL assignment statements.

The DEPOSIT and EXAMINE commands maintain a pointer to the current memory location. The EXAMINE command sets this pointer to the last location examined when you specify an EXAMINE command. You can refer to this location using the symbol "." in a subsequent EXAMINE or DEPOSIT command.

Description

When the EXAMINE command is executed, it displays the virtual memory address in hexadecimal and the contents in the radix requested as follows:

address: contents

If the address specified is not accessible to user mode, asterisks (****) are displayed in the contents field.

Radix Qualifiers: The radix default for a DEPOSIT or EXAMINE command determines how the commands interpret numeric literals, for example 256. The initial default radix is hexadecimal; all numeric literals in the command line are assumed to be hexadecimal values. If a radix qualifier modifies an EXAMINE command, that radix becomes the default for subsequent EXAMINE and DEPOSIT commands, until another qualifier overrides it. For example:

```
$ EXAMINE/DECIMAL 900
00000384:  0554389621
```

The EXAMINE command interprets the location 900 as a decimal number and displays the contents of that location in decimal. All subsequent DEPOSIT and EXAMINE commands assume that numbers you enter for addresses and data are decimal. Note that the EXAMINE command always displays the address location in hexadecimal.

Symbol names defined by = (Assignment Statement) commands are always interpreted in the radix in which they were defined.

Note that hexadecimal values entered as examine locations or as data to be deposited must begin with a numeric character (0 through 9). Otherwise, the command interpreter assumes that you have entered a symbol name and attempts symbol substitution.

You can use the radix operators %X, %D, or %O to override the current default when you enter the EXAMINE command, for example:

```
$ EXAMINE/DECIMAL %X900
00000900:  321446536
```

This command requests a decimal display of the data in the location specified as hexadecimal 900.

Length Qualifiers: The initial default length unit for the EXAMINE command is a longword. The EXAMINE command displays data one longword at a time, with blanks between longwords. If a length qualifier modifies the command, that length becomes the default length of a memory location for subsequent EXAMINE and DEPOSIT commands, until another qualifier overrides it.

Restriction on Placement of Qualifiers: The EXAMINE command analyzes expressions arithmetically. Therefore, qualifiers (which must be preceded by /) are interpreted correctly only when they appear immediately after the command name.

Command Qualifiers

/ASCII

Requests that the data at the specified location be displayed in ASCII.

Binary values that do not have ASCII equivalents are displayed as periods (.).

When you specify /ASCII, or ASCII mode is the default, hexadecimal becomes the default radix for numeric literals.

/BYTE

Requests that data at the specified location be displayed one byte at a time.

/DECIMAL
Requests that the contents of the specified location be displayed in decimal.

/HEXADECIMAL
Requests that the contents of the specified location be displayed in hexadecimal.

/LONGWORD
Requests that data at the specified location be displayed one longword at a time.

/OCTAL
Requests that the contents of the specified location be displayed in octal.

/WORD
Requests that data at the specified location be displayed one word at a time.

Examples

1.
```
$ RUN   MYPROG
^Y
$ EXAMINE   2678
0002678:  1F4C5026
$ CONTINUE
```

The RUN command begins execution of the image MYPROG. While MYPROG is running, CTRL/Y interrupts its execution, and the EXAMINE command requests a display of the contents of virtual memory location hexadecimal 2678.

2.
```
$ BASE = %X1C00
$ READBUF = BASE + %X50
$ ENDBUF = BASE + %XA0
$ RUN   TEST
^Y
$ EXAMINE/ASCII READBUF:ENDBUF
00001C50:  BEGINNING OF FILE MAPPED TO GLOBAL SECTION
    .
    .
```

Before executing the program MYPROG, symbolic names are defined for the program's base address, and for labels READBUF and ENDBUF; all are expressed in hexadecimal using the radix operator %X. READBUF and ENDBUF define offsets from the program base.

While the program is executing, CTRL/Y interrupts it and the EXAMINE command requests a display in ASCII of all data between the specified memory locations.

EXIT

Terminates processing of the current command procedure. If the command procedure was executed from within another command procedure, control returns to the calling procedure.

Format

```
EXIT   [status-code]

  Command Qualifiers

  None.
```

Prompts

None.

Command Parameters

status-code
: Defines a numeric value for the reserved global symbol $STATUS. The value can be tested by the next outer command level. The low-order three bits of the longword integer value change the value of the reserved global symbol $SEVERITY.

  If you do not specify a status code, the current value of $STATUS is not changed and control returns to the outer level with the status of the most recently executed command or program.

Description

When a command procedure returns control to the interactive command level, the command interpreter uses the current value of $STATUS to locate and display the system message associated with the value, if any. Note that even numeric values generally produce warning, error, and fatal error messages and that odd numeric values generally produce either no message or a success or informational message. Example 4, below, shows how to use the EXIT command to determine the message and symbolic error name associated with a hexadecimal status code.

The EXIT command performs no operation if issued at the interactive command level.

Examples

1.
```
$ ON WARNING THEN EXIT
$ FORTRAN 'P1'
$ LINK 'P1'
$ RUN 'P1'
```

The EXIT command is used as the target of an ON command; this statement ensures that the command procedure terminates whenever any warnings or errors are issued by any command in the procedure.

The procedure exits with the status value of the command or program that caused the termination.

2.
```
$ @SUBTEST
$ IF $STATUS .EQ. 7 THEN GOTO PROCESS
    .
    .
$ EXIT
$ PROCESS:
    .
    .
```

This procedure executes a second procedure, named SUBTEST.COM. When SUBTEST.COM completes, the outer procedure tests the value of the symbol $STATUS which may be set by SUBTEST as follows:

```
$ PATH1:
    .
    .
$ EXIT 7
$ PATH2:
    .
    .
$ EXIT 9
```

3.
```
$ IF P1.EQS."" THEN -
     INQUIRE P1 "Enter File-spec (null to exit)"
$ IF P1.EQS."" THEN EXIT
$ PRINT 'P1'/AFTER=20:00/COPIES=50/FORMS=6
```

A command procedure tests whether a parameter was passed to it; if not, it prompts for the required parameter. Then it retests the parameter P1. If a null string, indicated by a carriage return for a line with no data, is entered, the procedure exits. Otherwise, it executes the PRINT command with the current value of P1 as the input parameter.

4.
```
$ IF P1.EQS."" THEN INQUIRE P1 "Code"
$ CODE = %X'P1'
$ EXIT 'CODE'
```

This short command procedure illustrates how to determine the system message, if any, associated with a hexadecimal system status code. The procedure requires a parameter and prompts if none is entered. Then, it prefixes the value with the radix operator %X and assigns this string to the symbol CODE. Finally, it issues the EXIT command with the hexadecimal value. For example:

```
$ @E 1C
%SYSTEM-F-EXQUOTA, exceeded quota
```

When the procedure exits, the value of $STATUS is ^X1C, which causes the EXQUOTA message.

FORTRAN

Invokes the VAX-11 FORTRAN IV-PLUS[1] compiler to compile one or more source programs. This command is described in detail in the VAX-11 FORTRAN IV-PLUS User's Guide.

Format

```
FORTRAN   file-spec,...

  Command Qualifiers          Default

  None.

  File Qualifiers

  /[NO]CHECK[=option,...]     /CHECK=OVERFLOW
  /CONTINUATIONS=n            /CONTINUATIONS=19
  /[NO]DEBUG[=option]         /DEBUG=TRACEBACK
  /[NO]D_LINES                /NOD_LINES
  /[NO]I4                     /I4
  /[NO]LIST[=file-spec]
  /[NO]MACHINE_CODE           /NOMACHINE_CODE
  /[NO]OBJECT[=file-spec]
  /[NO]OPTIMIZE               /OPTIMIZE
  /[NO]WARNINGS               /WARNINGS
  /WORK_FILES=n               /WORK_FILES=2
```

Prompts

```
File:       file-spec,...
```

Command Parameters

file-spec,...

Specifies one or more FORTRAN source programs to be compiled. If you do not specify a file type, the compiler uses the default file type of FOR.

You can specify more than one input file. If you separate the file specifications with commas (,), each file is compiled separately. If you separate the file specifications with plus signs (+), the files are concatenated and compiled as a single input file, producing one object module and, if /LIST is specified, one listing.

---

[1] Available under separate license

File Qualifiers

/CHECK[=option,...]
/NOCHECK

Controls whether the compiler produces extra code to check for program correctness at run time. You can request the following options:

BOUNDS	Produce code to check that all array references are to addresses within the address boundaries specified in the array declaration.
OVERFLOW	Enable integer overflow traps. Fixed-point calculations involving BYTE, INTEGER*2, and INTEGER*4 data types are checked for arithmetic overflow.
ALL	Provide both BOUNDS and OVERFLOW checks. Note that /CHECK=ALL is equivalent to /CHECK.
NONE	Provide no checking. Note that /CHECK=NONE is equivalent to /NOCHECK.

By default, the compiler produces code to check only for integer overflow traps.

If you specify either /CHECK=BOUNDS or /CHECK=OVERFLOW, the other is implicitly canceled.

/CONTINUATIONS=n

Specifies the maximum number of continuation lines to be permitted.

The number of lines, n, is a decimal number from 0 through 99. By default, the compiler accepts a maximum of 19 continuation lines.

/DEBUG[=option]
/NODEBUG

Controls whether the compiler makes local symbol table and traceback information available to the debugger and the run time error reporting mechanism.

You can request the following options:

SYMBOLS	Provide the debugger with local symbol definitions for user-defined variables, arrays, labels on executable statements, and compiler-generated line numbers.
TRACEBACK	Provide compiler-generated line numbers so that the debugger and the run-time error traceback routine can translate virtual addresses into source program subroutine names and line numbers.
ALL	Provide both local symbol table and traceback information. Note that /DEBUG=ALL is equivalent to /DEBUG.
NONE	Do not provide either local symbol table or traceback information. Note that /DEBUG=NONE is equivalent to /NODEBUG.

By default, the compiler produces only an address correlation table.

If you specify either /DEBUG=SYMBOLS or /DEBUG=TRACEBACK, the other item is implicitly canceled.

For details on how to debug a FORTRAN program with the VAX-11 Symbolic Debugger, see the VAX-11 FORTRAN IV-PLUS User's Guide.

/D_LINES
/NOD_LINES

Indicates whether the compiler reads and compiles lines that have a D in column 1 of the source program. If you specify /D_LINES, lines that have a D in column 1 are compiled.

By default, the compiler assumes that lines beginning with a D are comments and does not compile them.

/I4
/NOI4

Controls how the compiler interprets INTEGER and LOGICAL declarations that do not specify a length. If you specify /NOI4, the compiler interprets these as INTEGER*2 and LOGICAL*2, respectively.

By default, the compiler interprets these declarations as INTEGER*4 and LOGICAL*4.

/LIST=[file-spec]
/NOLIST

Controls whether a listing file is produced.

If the FORTRAN command is executed from interactive mode, the compiler, by default, does not create a listing file. If the FORTRAN command is executed from batch mode, /LIST is the default; the compiler gives a listing file the same file name as the first input source file and a file type of LIS.

When you specify /LIST, you can control the defaults applied to the output file specification by the placement of the qualifier in the command, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers." The output file type always defaults to LIS.

/MACHINE_CODE
/NOMACHINE_CODE

Controls whether the listing produced by the compiler includes the machine language code generated by the compiler.

By default, the compiler does not include machine language code in the listing. The /MACHINE_CODE qualifier is ignored if /LIST is not specified, either explicitly or by default.

/OBJECT[=file-spec]
/NOOBJECT

Controls whether the compiler produces an output object module.

By default, the compiler produces an object module that has the same file name as the input source file and a file type of OBJ. When you specify /OBJECT, you can control the defaults applied to the output file specification by the placement of the qualifier in the command, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers." The output file type always defaults to OBJ.

/OPTIMIZE
/NOOPTIMIZE

Controls whether the compiler optimizes the compiled program to generate more efficient code.

Use /NOOPTIMIZE in conjunction with the /DEBUG qualifier to link a FORTRAN program with the debugger so that variables always contain their updated values.

/WARNINGS
/NOWARNINGS

Controls whether the compiler produces diagnostic messages for warning conditions.

Use /NOWARNINGS to override the compiler default, which is to issue warning diagnostic messages.

/WORK_FILES=n

Specifies the number of work files that the compiler should use during compilation.

You can specify 1, 2, or 3. By default, the compiler uses two work files. Increasing the number of work files enables larger programs to be compiled, but may decrease the speed of the compilation.

Examples

1.
```
$ FORTRAN SCANLINE
```

The FORTRAN command compiles the program SCANLINE.FOR and produces an object module SCANLINE.OBJ. If this command is executed in a batch job, the compiler also creates a listing file named SCANLINE.LIS.

2.
```
$ FORTRAN A+B/LIST, C+D/LIST=ALL/OBJECT=ALL
```

This FORTRAN command requests two separate compilations. For the first compilation, the FORTRAN command concatenates the files A.FOR and B.FOR to produce an object module named A.OBJ and a listing file named B.LIS.

For the second compilation, the FORTRAN command concatenates the files C.FOR and D.FOR and compiles them to produce an object module named ALL.OBJ and a listing named ALL.LIS.

3.
```
$ FORTRAN/NOOPTIMIZE/DEBUG  SCAN/OBJECT=DBGSCAN
$ LINK/DEBUG DBGSCAN
$ RUN DBGSCAN

     DEBUG, Version 1.0

%DEBUG-I-INITIAL, language is FORTRAN, scope and module set
to 'SCAN'

DBG>
```

The FORTRAN command compiles the source program SCAN with the debugger, including both the symbol table information and traceback information; the /OBJECT qualifier requests that the object module be named DBGSCAN. The LINK command links the debugger with the object module and the RUN command executes the image. When the debugger prompts, you can begin entering DEBUG commands.

GOTO

Transfers control to a labeled statement in a command procedure.

Format

```
GOTO   label

  Command Qualifiers:

  None.
```

Prompts

Label: label

Command Parameters

label

Specifies a 1- to 255-alphanumeric character label appearing as the first item on a command line. When the GOTO command is executed, control passes to the command following the specified label.

The label can precede or follow the GOTO statement in the current command procedure. It must be terminated with a colon (:) and may not contain embedded blanks.

Description

Use the GOTO command in command procedures to transfer control to a line that is not the next line in the procedure. If the command stream is not being read from a random access device (that is, a disk device), the GOTO command performs no operation.

The command interpreter does not check for duplicate labels. The following rules apply:

- If duplicate labels precede and follow the GOTO command, control is given to the label preceding the command.
- If duplicate labels all precede the GOTO command, control is given to the most recent label, that is, the one nearest the GOTO command.
- If duplicate labels all follow the GOTO command, control is given to the one nearest the GOTO command.

If a label does not exist in the current command procedure, the procedure cannot continue and is forced to exit.

Examples

1.
```
$ IF P1.EQS."HELP" THEN GOTO TELL
$ IF P1.EQS."" THEN GOTO TELL
   .
   .
$ EXIT
$ TELL:
$ TYPE SYS$INPUT
   .
   .
```

The IF command checks the first parameter passed to the command procedure; if this parameter is the string HELP or is not specified, the GOTO command is executed, and control is passed to the line labeled TELL. Otherwise, the procedure continues executing until the EXIT commmand is encountered. At the label TELL, a TYPE command displays data in the input stream that documents how to use the procedure.

2.
```
$ ON ERROR THEN GOTO CHECK
   .
   .
   .
$ GOTO END
$ CHECK:
   .
   .
$ END:
$ EXIT
```

The ON command establishes an error handling routine. If any command or procedure subsequently executed in the command procedure returns an error or severe error return, the GOTO command transfers control to the label CHECK.

HELP

Displays on the terminal information available in the system HELP files.

Format

```
HELP   [keyword [keyword]...]

  Command Qualifiers

  None.
```

Prompts

None.

Command Parameters

keyword [keyword]...
Specifies one or more keywords that indicate what information you want. Information is located in a hierarchical manner, depending on the level of information required. The levels are:

1. <NULL> - describes the HELP command, and lists keywords you can specify to obtain information about commands and programs that are documented. Each item in this list is a keyword in the second level of the hierarchy. Most keywords are names of DCL commands, but other information is provided. For example, the keyword SPECIFY is at the first level of a hierarchy of information.

2. Command-name or program-name - describes the command function and format and lists additional information available. This list of information provides keywords for the next level.

3. Command name or program-name followed by a specific item - provides descriptions of command parameters, or particular keywords or qualifiers.

If you specify a wild card in the place of any parameter, the HELP command displays all information available at that level.

Examples

1. `$ HELP ASSIGN`

   The HELP command displays an abstract of the ASSIGN command function, its format, and lists the keyword options you can type to obtain more information.

2. `$ HELP ASSIGN PARAMETERS`

   The HELP command displays a description of the parameters for the ASSIGN command.

3. `$ HELP SPECIFY LEXICAL FUNCTIONS`

   The HELP command lists the lexical command functions.

4. `$ HELP PRINT /AFTER`

   The HELP command lists the information available about the /AFTER qualifier of the PRINT command.

5. `$ HELP SUBMIT *`

   The HELP command displays information about the SUBMIT command available at the second level, that is, the information that would be displayed if you specified each of the keywords listed when you issued HELP SUBMIT.

6. `$ HELP SET TERMINAL`

   The HELP command lists information about the TERMINAL keyword option of the SET command.

IF

Tests the value of an expression and executes a command if the test is true. Any arithmetic or logical expression is considered true if the result of the expression is an odd numeric value; an expression is false if the result is an even value.

Format

```
IF expression THEN [$] command

  Command Qualifiers

  None.
```

Prompts

None.

Command Parameters

expression

Defines the test to be performed. The expression can consist of one or more numeric constants, string literals, or symbolic names separated by logical, arithmetic, or string operators.

The logical, arithmetic, and string operators are listed below. Combinations of operations are evaluated in the order of precedence indicated, where 1 is the lowest precedence and 6 is the highest. You can use parentheses to group items in an expression to override the order in which operations are evaluated.

Operator	Precedence	Operation
.OR.	1	Logical sum
.AND.	2	Logical product
.NOT.	3	Logical complement
.EQ.	4	Arithmetic equal to
.GE.	4	Arithmetic greater than or equal to
.GT.	4	Arithmetic greater than
.LE.	4	Arithmetic less than or equal to
.LT.	4	Arithmetic less than
.NE.	4	Arithmetic not equal to
.EQS.	4	String equal to
.GES.	4	String greater than or equal to
.GTS.	4	String greater than
.LES.	4	String less than or equal to
.LTS.	4	String less than
.NES.	4	String not equal to
+	5	Arithmetic sum
-	5	Arithmetic difference
*	6	Arithmetic product
/	6	Arithmetic quotient

For further details on the syntax requirements and valid ways of specifying expressions, see Section 6.6 "Rules for Forming Expressions."

command
Defines the action to take if the result of the expression is true.

You can specify any valid DCL command following the keyword THEN. Optionally, you can precede the command with a dollar sign.

Examples

1.
```
$ COUNT = 0
$ LOOP:
$ COUNT = COUNT + 1
    .
    .
$ IF COUNT.LE.10 THEN GOTO LOOP
$ EXIT
```

This example shows how to establish a loop in a command procedure using a symbol named COUNT and an IF statement that checks the value of COUNT and performs an EXIT command when the value of COUNT is greater than 10.

2.
```
$ IF P1.EQS."" THEN GOTO DEFAULT
$ IF P1.EQS."A" .OR. P1.EQS."B" THEN GOTO 'P1'
$ WRITE SYS$OUTPUT "Unrecognized parameter option ''P1' "
$ EXIT
$ A: !  Process option a
    .
    .
$ EXIT
$ B: !  Process option b
   .
   .
$ EXIT
$ DEFAULT: !  Default processing
    .
    .
$ EXIT
```

This example shows a command procedure that tests whether a parameter was passed; if not, the IF statement assigns a value to P1 to provide a default processing option. The GOTO command passes control to the label specified as the parameter.

If the procedure is executed with a parameter, the procedure uses that parameter to determine the label to branch to. For example:

```
@TESTCOM A
```

When the procedure executes, it determines that P1 is not null, and branches to the label A. Note that the EXIT command causes an exit from the procedure before the label B.

3.
```
$ SET NOON
    .
    .
$ LINK CYGNUS,DRACO,SERVICE/LIBRARY
$ IF .NOT.$STATUS THEN EXIT
$ RUN CYGNUS
```

A command procedure uses the SET NOON command to disable error checking by the command procedure. Then, the IF command is used following the execution of a LINK command to test the value of the reserved global symbol $STATUS. If the LINK command returns an error status value, the command procedure exits.

INITIALIZE

Formats and writes a label on a mass storage volume.

Format

```
INITIALIZE   device-name[:]   volume-label

  Command Qualifiers[1]                    Default

  /OWNER_UIC=uic
  /PROTECTION=code

  Qualifiers for Tapes

  /DENSITY=n
  /OVERRIDE=options,...

  Qualifiers for Disks

  /ACCESSED=n                              /ACCESSED=3
  /BADBLOCKS=list
  /CLUSTER_SIZE=n
  /DATA_CHECK[=options,...]
  /DIRECTORIES=n                           /DIRECTORIES=16
  /EXTENSION=n                             /EXTENSION=5
  /FILE_PROTECTION=code
  /GROUP
  /HEADERS=n                               /HEADERS=16
  /INDEX=position                          /INDEX=MIDDLE
  /MAXIMUM_FILES=n
  /[NO]SHARE                               /SHARE
  /STRUCTURE=level                         /STRUCTURE=2
  /SYSTEM
  /USER_NAME=string
  /[NO]VERIFIED                            /VERIFIED
  /WINDOWS=n                               /WINDOWS=7
```

Prompts

Device: device-name

Label: volume-label

Command Parameters

device-name

Specifies the name of the device on which the volume to be initialized is physically mounted.

The device does not have to be currently allocated; however, this is the recommended practice.

---

[1] For convenience, qualifiers that are applicable only to disks and to tapes are listed separately. All qualifier descriptions appear in alphabetical order, however.

volume-label
Specifies the identification to be encoded on the volume. For a disk volume, you can specify a maximum of 12 alphanumeric characters; for a tape volume, you can specify a maximum of 6 alphanumeric characters.

Description

The default format for disk volumes in the VAX/VMS operating system is called the Files-11 Structure Level 2. The default for tape volumes is ANSI standard labels, ANSI X3.27-1977, level 3.

The INITIALIZE command can also initialize volumes in the Files-11 Structure Level 1 format.

You do not need any special privileges to initialize a blank disk or tape volume. If a volume has previously been written, however, your UIC must match the owner UIC on the volume, the protection code on the volume must allow you write access, or you must have the user privilege to override volume protection (VOLPRO).

For examples of initializing and using disks and tapes, see Chapter 3, "Disk and Tape Volumes."

Many of the INITIALIZE command qualifiers allow you to specify parameters that can maximize input/output efficiency. For information on these parameters, see the Introduction to VAX-11 Record Management Services and VAX-11 Record Management Services Reference Manual.

Command Qualifiers

/ACCESSED=n
Specifies, for disk volumes, the number of directories to be maintained in system space for ready access.

The user privilege OPER is required to use the /ACCESSED qualifier. Legal values for n are 0 through 255. If /ACCESSED is not specified, the INITIALIZE command uses the default value of 3.

/BADBLOCKS=list
Specifies, for disk volumes, specific areas on the volume that are faulty. The INITIALIZE command marks the areas as allocated so that no data will be written in them.

You can specify one or more areas, using either or both of the formats shown below. If you specify more than one area, separate the specifications with commas and enclose the list in parentheses.

lbn[:count] Specify a logical block number on the disk volume, and optionally a count of logical blocks beginning with the logical block specified, to be marked allocated

sector.track.cyl[:count]
Specify a specific sector, track, and cylinder on the disk volume, and optionally a count of blocks, beginning with the first block specified, to be marked allocated

Use of this qualifier is device-dependent. It is not required for RK06/7 or RM03 disks; nor is it required for disks that have been scanned for bad block data with the BAD utility program.

For information on how to run the BAD utility, see the VAX/VMS Operator's Guide.

/CLUSTER_SIZE=n

Defines, for disk volumes, the minimum allocation unit, in blocks. The maximum size you can specify for a volume is 1/100 the size of the volume; the minimum size you can specify is calculated with the formula:

$$\frac{\text{disk size}}{255*4096}$$

If not specified, the INITIALIZE command uses the following default values:

Device	Default
RK06	2
RK07	4
RM03	6
RP04/5	6
RP06	11

The default value of structure level 1 disks is always 1.

/DATA_CHECK[=options,...]

Defines a default for data check operations following all reads and/or writes to the volume. You can specify either or both of the following options:

READ	Perform checks following all read operations
WRITE	Perform checks following all write operations

If you specify /DATA_CHECK without specifying an option, the system assumes /DATA_CHECK=WRITE. If you do not specify /DATA_CHECK, the system performs no checking; this is the default. You can override the checking you specify at initialization for disks when you issue a MOUNT command to mount the volume.

/DENSITY=n

Specifies, for tape volumes, the density in bits per inch (bpi) at which the tape is to be written. You can specify a density of either 800 or 1600.

If you do not specify a density for a blank tape, the system uses a default density of 1600. If you do not specify a density for a tape that was previously written, the system uses the density at which the tape was last written.

/DIRECTORIES=n

Specifies, for disk volumes, the number of entries to preallocate for user directories.

The legal values are in the range of 16 through 16000; if you do not specify a value, the INITIALIZE command uses the default value of 16.

/EXTENSION=n

Specifies, for disk volumes, the number of blocks to use as a default extension size for all files on the volume. The extension default is used when a file increases to a size greater than its initial default allocation during an update.

You can specify a value in the range of 0 through 65535; if you do not specify a default extension size, the INITIALIZE command uses a value of 5.

/FILE_PROTECTION=code

Defines, for disk volumes, the default protection to be applied to all files on the volume.

Specify the code according to the standard syntax rules for specifying protection. (These rules are given in the discussion of the SET PROTECTION command.) Any attributes not specified are taken from the current default protection.

Note that this attribute is not used when the volume is being used on a VAX/VMS system, but is provided to control the use of the volume on RSX-11M systems. VAX/VMS always uses the default file protection; the protection can be changed with the SET PROTECTION/DEFAULT command.

/GROUP

Defines a disk volume as a group volume. The owner UIC of the volume defaults to the group number of the user issuing the command and a member number of 0.

This qualifier defines the volume protection as RWED for the system, owner and group.

/HEADERS=n

Specifies, for disk volumes, the number of file headers to be allocated initially for the index file. The minimum value you can specify is 16; the maximum value is the value set with the /MAXIMUM_FILES qualifier.

By default, the INITIALIZE command allocates 16 file headers.

/INDEX=position

Requests, for disk volumes, that the index file for the volume's directory structure be placed in a specific location on the volume.

You can specify one of the following options:

BEGINNING	Place the index file at the beginning of the volume
MIDDLE	Place the index file in the middle of the volume
END	Place the index file at the end of the volume
BLOCK:n	Place the index file at the beginning of the logical block specified

By default, the INITIALIZE command places the index file in the middle of the volume.

/MAXIMUM_FILES=n

Restricts, for disk volumes, the maximum number of files that the volume can contain, overriding the default value. The defaults are:

Device	Default File Maximum
RK06	4000
RK07	8000
RP04/RP05	15000
RP06	25000
RM03	15000
Floppy	123

The maximum size you can specify for any volume is the volume size, in blocks, divided by (cluster factor plus 1); the minimum value is 0. Note, however, that you should specify a low file maximum only after careful consideration. Once set, the maximum can only be increased by re-initializing the volume.

/OVERRIDE=options,...

Requests the INITIALIZE command to ignore data on a tape volume that protects it from being overwritten. You can specify one or both of the following options:

EXPIRATION — Override the expiration date on the volume (the date is indicated by the expiration date of the first file on the volume)

ACCESSIBILITY — Override a non-blank accessibility field in the VOL1 or HDR1 label (this field is never set by VAX/VMS, but may be set by other operating systems)

You must be the owner of the tape volume or have the user privilege to override volume protection (VOLPRO) in order to initialize a tape that has not reached its expiration date or has a non-blank accessibility field.

/OWNER_UIC=uic

Specifies the user identification code to be assigned ownership of the volume and of system files on the volume. Specify the UIC in the format:

[g,m]

g is an octal number in range 0 through 377 representing the group number.
m is an octal number in the range 0 through 377 representing the member number.

The brackets are required.

If you do not specify /OWNER_UIC, your current UIC is assigned ownership of the volume.

/PROTECTION=code

Specifies the protection to be applied to the volume. The protection controls who can read, write, create, and delete files on the volume. If you do not specify a protection code, protection defaults to all access to all categories of user. Note that the /GROUP, /SHARE, and /SYSTEM qualifiers can also be used to define protection for disk volumes.

Specify the code according to the standard syntax rules for specifying protection. (These rules are given in the discussion of the SET PROTECTION command.) Any attributes not specified default to no access.

When you specify a protection code for an entire disk volume, access type E (execute) indicates create access.

The system only applies read and write access restrictions with respect to tapes; create and delete access are meaningless. Moreover, the system and the owner are always given both read and write access to tapes, regardless of what you specify in a protection code.

/SHARE
/NOSHARE

Controls whether a disk volume is shareable. The protection code for the volume defaults to all types of access for all categories of user. If you specify /NOSHARE, the protection code defaults to no access for group and world.

/STRUCTURE=level

Specifies, for disk volumes, whether the volume should be formatted in Files-11 Structure Level 1 or level 2. By default, disk volumes are formatted in Files-11 Structure level 2.

If you specify /STRUCTURE=1, the /CLUSTER and /DATA_CHECK qualifiers are not allowed. The default protection for a structure level 1 disk is all types of access to system, owner, and group, and read access to all other users.

/SYSTEM

Defines a disk volume as a system volume. The owner UIC of the volume defaults to [1,1] and default protection provides all types of access to the volume to all users.

No user privilege is required to use the /SYSTEM qualifier; however, only users with system UICs can create directories on system volumes.

/USER_NAME=string

Specifies, for disk volumes, a user name of up to 12 characters to be recorded on the volume. If not specified, the INITIALIZE command uses the user name under which you logged in.

/VERIFIED
/NOVERIFIED

Indicates, for disk volumes, whether the disk has bad block data on it. /VERIFIED is the default; the INITIALIZE command assumes that disks contain bad block data and uses the data to mark the bad blocks as allocated. Use /NOVERIFIED to request INITIALIZE to ignore bad block data on the disk.

/WINDOWS=n

Specifies, for disk volumes, the number of mapping pointers to be allocated for file windows. When a file is opened, the file system uses the mapping pointers to access data in the file. You can specify a value in the range of 7 through 80. The default number of pointers is 7.

Examples

1.
```
$ ALLOCATE DMA2:  TEMP
  _DMA2: ALLOCATED
$ INITIALIZE  TEMP  BACK_UP_FILE
$ MOUNT  TEMP  BACK_UP_FILE
%MOUNT-I-MOUNTED, BACK_UP_FILE mounted on _DMA2:
$ CREATE/DIRECTORY  TEMP:[ARCHIE]
$ COPY *.*  TEMP:[ARCHIE]
```

The above sequence of commands shows how to initialize an RK06/7 volume for backup. First, the device is allocated, to ensure that no one else can access it. Then, when the volume is physically mounted on the device, the INITIALIZE command initializes it. When the volume is initialized, the MOUNT command makes the file structure available. Before you can place any files on the volume, you must create a directory, as shown in the CREATE command example. Finally, the COPY command copies the highest existing versions of files on the default disk to the backup disk.

2.
```
$ ALLOCATE MT:
  _MTB1:  ALLOCATED
$ INITIALIZE MTB1:  SOURCE
$ MOUNT MTB1:  SOURCE
%MOUNT-I-MOUNTED, SOURCE mounted on _MTB1:
$ COPY *.FOR  MTB1:
$ DIRECTORY MTB1:
    .
    .
$ DISMOUNT MTB1:
```

These commands show the procedure necessary to initialize a tape. After allocating a drive, the tape is loaded on the device and the INITIALIZE command writes the label SOURCE on it. Then, the MOUNT command mounts the tape so that files can be written on it.

INQUIRE

Requests interactive assignment of a value for a local or global symbol during the execution of a command procedure.

Format

```
INQUIRE   symbol-name   [prompt-string]

  Command Qualifiers          Default

  /GLOBAL                     /LOCAL
  /LOCAL                      /LOCAL
```

Prompts

None.

Command Parameters

symbol-name
Specifies a 1 - to 255-alphanumeric character symbol to be given a value.

prompt-string
Specifies the prompt to be displayed at the terminal when the INQUIRE command is executed. If the prompt string contains any lowercase characters or multiple blanks or tabs, enclose it in quotation marks.

When the system displays the prompt string at the terminal, it places a colon (:) at the end of the string.

If you do not specify a prompt string, the command interpreter uses the symbol name to prompt for a value.

Description

The INQUIRE command displays the prompting message to and reads the response from the device SYS$COMMAND. This means that when the INQUIRE command is executed in a command procedure executed interactively, the prompting message is always displayed on the terminal, regardless of the level of nesting of command procedures. When an INQUIRE command is issued in a batch job, the command reads the response from the next line in the command procedure; if procedures are nested, it reads the response from the first level command procedure.

Command Qualifiers

/GLOBAL
Specifies that the symbol be placed in the global symbol table.

/LOCAL
Specifies that the symbol be placed in the local symbol table for the current command procedure.

This is the default.

Examples

1.
```
$ INQUIRE CHECK "Enter Y[ES] to continue"
$ IF .NOT.CHECK THEN EXIT
```

The INQUIRE command displays the following prompting message at the terminal:

```
Enter Y[ES] to continue:
```

The IF command tests the value entered. If you enter an odd numeric value or any non-quoted character string that begins with either a "T" or a "Y" the symbol CHECK will be considered true and the procedure will continue executing. If you enter an even numeric value, any nonquoted character string that begins with either an "N" or an "F," or a null string, the symbol will be considered false and the procedure will exit.

2.
```
$ INQUIRE COUNT
$ IF COUNT.GT.10 THEN GOTO SKIP
    .
    .
$ SKIP:
```

The INQUIRE command prompts for a count with the message:

```
COUNT:
```

Then, the command procedure uses the value of the symbol COUNT to determine whether to execute the next sequence of commands or to transfer control to the line labeled SKIP.

3.
```
$ IF P1.EQS."" THEN INQUIRE P1 FILE NAME
$ FORTRAN 'P1'
```

The IF command checks whether a parameter was passed to the command procedure by checking if the symbol P1 is null; if it is, it means that no parameter was specified, and the INQUIRE command is issued to prompt for the parameter. If P1 was specified, the INQUIRE command is not executed, and the FORTRAN command compiles the name of the file specified as a parameter.

JOB

Identifies the beginning of a batch job submitted through a system card reader.

Format

```
JOB   user-name

  Qualifiers                      Default

  /AFTER=absolute-time
  /[NO]DELETE                     /DELETE
  /NAME=job-name                  /NAME=INPBATCH
  /PARAMETERS=parameters,...
  /PRIORITY=n
  /QUEUE=queue-name               /QUEUE=SYS$BATCH
  /[NO]TRAILING_BLANKS            /TRAILING_BLANKS
```

Prompts

None.

Command Parameters

user-name
Identifies the user name under which the job is to be run. Specify the user name just as you would enter it during the login procedure. All qualifiers must follow the user-name; otherwise the job is not submitted.

Description

All batch jobs submitted to the system through the system card reader must be preceded by a JOB card. The $ is required. The JOB card identifies the user submitting the job, and must be followed by a PASSWORD card giving the password.

The user name and password are validated using the system authorization file in the same manner as they are validated in the login procedure. The process that executes the batch job is assigned the disk and directory defaults and privileges associated with the account. If a LOGIN.COM file exists, it is executed at the start of the job.

The end of a batch job is signaled by the EOJ command, by an EOF card (12-11-0-1-6-7-8-9 overpunch), or by another JOB card.

Command Qualifiers

/AFTER=absolute-time
Requests that the job be held until after a specific time.

Specify the time value according to the rules for entering absolute times (these rules are given in Section 6.8).

If the specified time has already passed, the job is queued for immediate processing.

/DELETE
/NODELETE
Controls whether the batch input file is saved after the job is processed. By default, the job, which is saved for the duration of the job, is deleted after processing. If you specify /NODELETE, the file is saved under the name INPBATCH.COM, by default. If you specify the /NAME qualifier, the file name of the file is the same as the name you specify with /NAME.

/NAME=job-name
Specifies a 1- to 8-alphanumeric character file name string to be used as the job name and as the file name for the batch job log file. By default, the system gives the output log file a file name of INPBATCH.

/PARAMETERS=parameters,...
Specifies from 1 to 8 optional parameters to be passed to the command procedure. The parameters define values to be equated to the symbols named P1, P2, P3, and so on, in the batch job. The symbols are local to the initial input stream.

If you specify more than one parameter, separate them with commas and enclose them in parentheses.

The commas delimit the parameters. To specify a parameter that contains any special characters or delimiters, enclose the parameter in quotation marks.

/PRIORITY=n
Specifies the priority for the specified job.

The priority, n, must be in the range of 0 through 31, where 0 is the lowest priority and 31 is the highest.

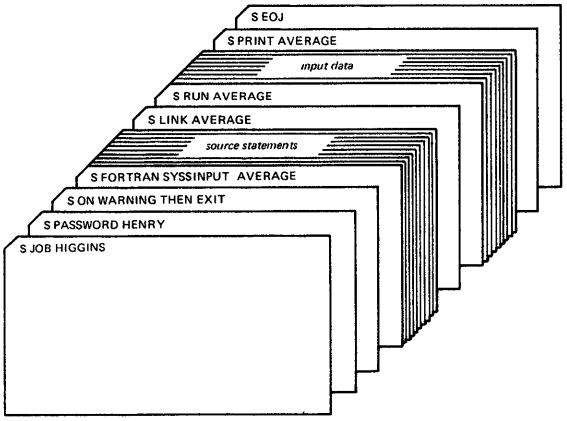
By default, jobs are assigned the same priority as your current process's base priority; the user privilege ALTPRI is required to set a priority value that is higher than your current process's base priority. If you specify a higher priority than your current base priority and you do not have this privilege, the job is given your current base priority, by default.

/QUEUE=queue-name
Specifies the name of a particular batch job queue in which the job is to be entered. If you do not specify /QUEUE, then the job is placed in the default system batch job queue, SYS$BATCH.

/TRAILING_BLANKS
/NOTRAILING_BLANKS
Controls whether input cards in the card deck are read in card image form or if input records are truncated at the last non-blank character. By default, the system does not strip trailing blanks from the records read through the card reader. Use the /NOTRAILING_BLANKS qualifier to request that input records be truncated.

Examples

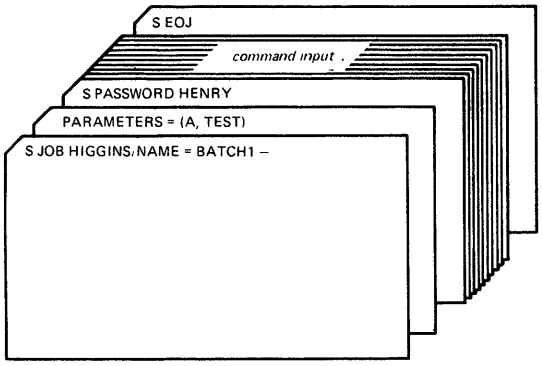
1.



The JOB and PASSWORD cards identify and authorize the user HIGGINS to enter batch jobs. The command stream consists of a FORTRAN command and FORTRAN source statements to be compiled. The file name AVERAGE following the device name SYS$INPUT provides the compiler with a file name for the object and listing files. The output files are cataloged in the user HENRY's default directory.

If the compilation is successful, the LINK command creates an executable image, and the RUN command executes it. Input for the program follows the RUN command in the command stream. The last command in the job prints the program listing.

2.



The /NAME qualifier on the JOB card specifies a name for the batch job. When the job completes, the printed log file will be identified as BATCH1.LOG. The JOB card is continued onto a second line with the continuation character (-). The /PARAMETERS qualifier defines P1 as A and P2 as TEST.

LIBRARY

Creates or modifies an object module library or a macro library; or inserts, deletes, replaces, or lists modules, macros, or global symbol names in a library.

Format

```
LIBRARY library [file-spec,...]

   Command Qualifiers          Default

   /COMPRESS[=options,...]     /REPLACE
   /CREATE[=options,...]       /REPLACE
   /DELETE=module,...          /REPLACE
   /EXTRACT[=module,...]
   /FULL
   /[NO]GLOBALS                /GLOBALS
   /INSERT                     /REPLACE
   /[NO]LIST[=file-spec]       /NOLIST
   /MACRO                      /OBJECT
   /[NO]NAMES                  /NONAMES
   /OBJECT                     /OBJECT
   /OUTPUT=file-spec
   /REMOVE=symbol,...
   /REPLACE                    /REPLACE
   /RSX11
   /SELECTIVE_SEARCH
   /SQUEEZE
```

Prompts

Library: library

File: file-spec

Command Parameters

library
Specifies the name of the library to be created or modified.

Wild cards are not allowed in the library file specification. If the file specification does not include a file type, the LIBRARY command assumes a default file type of OLB if /OBJECT is specified either explicitly or by default; or a default file type of MLB if /MACRO is specified.

file-spec,...
Specifies the names of one or more files that contain modules to be inserted in the specified library. This parameter is required when you specify /INSERT or /REPLACE; it is optional when you specify /CREATE. Note that the default operation for the LIBRARY command is /REPLACE; if you do not specify a qualifier that requests a specific operation, you must enter the file-spec parameter.

If you do not specify any of the qualifiers /REPLACE, /INSERT, or /CREATE, the file-spec parameter is invalid.

If you specify more than one file, you can separate the file specifications with either plus signs (+) or commas (,). In either case, the contents of each file are inserted in the specified library.

No wild cards are allowed in the input file specifications. If any file specification does not include a file type, the LIBRARY command assumes a default file type of OBJ when /OBJECT is specified either implicitly or by default; a file type of MAR when /MACRO is specified and /RSX11 is not specified; and a file type of MAC when /MACRO and /RSX11 are specified.

Description

Libraries are files that contain one or more entries, or modules, of a similar type; and directories, or tables, that indicate the locations of individual modules within the library. There are two types of library:

- Object module libraries that catalog frequently called routines; you can use object module libraries as input to the linker. The linker searches the object module library when it encounters a reference it cannot resolve from the input files specified.

- Macro libraries that catalog macro definitions; you can use macro libraries as input to the assembler. The assembler searches the macro library when it encounters a macro that is not defined in the input file.

Both object module libraries and macro libraries have a directory called a module name table (MNT) that lists the modules (or, in the case of macro libraries, the macros) in the library. An object module library also contains a global symbol table (GST) that lists the global symbols defined in each of the modules in the library.

When the LIBRARY command adds a file to an object module library, it catalogs the module according to its module name, and not by the file name of the file containing the module.

The LIBRARY command creates libraries and modifies their contents. You can use DCL commands to manipulate a library in its entirety, for example, the DELETE, COPY, and RENAME commands can delete, make copies of, or rename libraries, respectively.

When you use the LIBRARY command, you can specify qualifiers that request more than one function in a single command, with some restrictions. The qualifiers that perform LIBRARY functions, related qualifiers, and qualifier incompatibilities are summarized in Table 1.

Table 1
LIBRARY Command Qualifiers

Qualifier	Related Qualifiers	Incompatible Functions
/COMPRESS	/OUTPUT	/CREATE, /EXTRACT
/CREATE[1]	/SQUEEZE[2], /GLOBALS[3] /SELECTIVE_SEARCH[3]	/COMPRESS, /EXTRACT
/DELETE	—	/EXTRACT
/EXTRACT	/OUTPUT	/COMPRESS, /CREATE, /DELETE /INSERT, /LIST, /REMOVE, /REPLACE
/INSERT	/SQUEEZE[2], /GLOBALS[3] /SELECTIVE_SEARCH[3]	/EXTRACT
/LIST	/FULL, /NAMES[3]	/EXTRACT
/REMOVE[3]	--	/EXTRACT
/REPLACE	/SQUEEZE[2], /GLOBALS[3] /SELECTIVE_SEARCH[3]	/EXTRACT

1 The /CREATE, /INSERT, and /REPLACE qualifiers are not incompatible; however, if more than one is specified, /CREATE takes precedence over /INSERT and /INSERT takes precedence over /REPLACE. The related qualifiers for /CREATE are applicable only if you enter one or more input files.

2 Indicates a qualifier that applies only to macro libraries

3 Indicates a qualifier that applies only to object module libraries

Command Qualifiers

/COMPRESS[=options,...]

Requests the LIBRARY command to recover unused space in the library resulting from module deletion. When you specify /COMPRESS, the LIBRARY command by default creates a new library with a version number one higher than the existing library. Use the /OUTPUT qualifier to specify an alternate name for the compressed library.

You can optionally specify one or more of the following options to increase or decrease the size of the library, overriding the values specified when the library was created:

BLOCKS:n Specify the number of 512-byte blocks to be allocated for the library

GLOBALS:n Specify the maximum number of global symbols the library can contain (for object module libraries only)

MODULES:n Specify the maximum number of modules or macros the library can contain

If you specify more than one option, separate them with commas or plus signs and enclose the list in parentheses.

/CREATE[=options,...]

Requests the LIBRARY command to create a new library. When you specify /CREATE, you can optionally specify a file or a list of files that contain modules to be placed in the library.

By default, the LIBRARY command creates an object module library; specify /MACRO to indicate that the library is a macro library.

Specify one or more of the following options to control the size of the library, overriding the system defaults:

BLOCKS:n Specify the number of 512-byte blocks to be allocated for the library. By default, the LIBRARY command allocates 100 blocks for a new library.

GLOBALS:n Specify the maximum number of global symbols the library can contain. By default, the LIBRARY command sets a maximum of 275 global symbols for an object module library. (A macro library does not have a global symbol directory; therefore the maximum for macro libraries defaults to 0.)

If you specify /RSX11, the default maximum number of global symbol table entries is 128.

MODULES:n Specify the maximum number of modules the library can contain. By default, the LIBRARY command sets a maximum of 75 modules for an object module library and 275 modules for a macro library. If you specify the MODULES option, the maximum value allowed for n is 4096.

If you specify /RSX11, the default maximum number of modules for an object module library is 512; the default maximum number of macros for macro libraries is 256.

If you specify more than one option, separate them with commas or plus signs and enclose the list in parentheses.

/DELETE=module,...

Requests the LIBRARY command to delete one or more modules from a library. You must specify the names of one or more modules to be deleted from the library. If you specify more than one module, separate them with commas or plus signs and enclose the list in parentheses.

When the LIBRARY command deletes modules from a library, it issues the message:

```
MODULES DELETED:
```

Then, it lists the names of modules it has successfully deleted.

The LIBRARY command does not physically remove a module from a library, but rather deletes its entry in the module name table. Use the /COMPRESS qualifier to compress a library from which modules have been deleted.

/EXTRACT=module,...

Copies one or more modules from an existing library into a new file. If you specify more than one module, separate the module names with commas or plus signs and enclose the list in parentheses.

If you specify the /OUTPUT qualifier in conjunction with /EXTRACT, the LIBRARY command writes the output into the file specified by the /OUTPUT qualifier. If you specify /EXTRACT and do not specify /OUTPUT, the LIBRARY command writes the file into a file that has the same file name as the library and a file type of OBJ, MAR, or MAC, depending on the /OBJECT, /MACRO, and /RSX11 qualifiers.

If you specify /EXTRACT in conjunction with /RSX11, then the module name on the /EXTRACT qualifier is optional: all modules in the specified library are concatenated into a single file with a file type of OBJ or MAC, depending on the /OBJECT and /MACRO qualifiers.

If you do not specify /RSX11, you must specify at least one module name.

/FULL

Requests a full description of each module in the module name table. Use this qualifier in conjunction with the /LIST qualifier to request LIBRARY to list each module in the library in the format:

```
entry SIZE:nnnnn INSERTED:dd-mmm-yyyy IDENT:nn
```

/GLOBALS
/[NO]GLOBALS

Controls, for object module libraries, whether the names of global symbols in modules being inserted in the library are included in the global symbol table.

By default, the LIBRARY command places all global symbol names in the global symbol table. Use /NOGLOBALS when you do not want the global symbol names in the global symbol table.

/INSERT

Requests the LIBRARY command to add the contents of one or more files to an existing library. If an object module file specified as input consists of concatenated object modules, the LIBRARY command creates a separate entry for each object module in the file; each module name table entry reflects an individual module name. If a macro file specified as input contains more than one macro definition, the LIBRARY command creates a separate entry for each macro, naming the module name table entries according to the names specified on the .MACRO directives.

When the LIBRARY command inserts modules into an existing library, it checks the module name table before inserting each module. If a module name, macro name, or global symbol name already exists in the library, the command issues an error message and does not add the module to the library. One or more modules may be successfully inserted before the error occurs.

To insert or replace a module in a library regardless of whether there is a current entry with the same name, use the /REPLACE qualifier.

/LIST[=file-spec]
/NOLIST

Controls whether or not the LIBRARY command creates a listing of the contents of the library.

/NOLIST is the default. If you specify /LIST without including a file specification, the LIBRARY command writes the output file to the current SYS$OUTPUT device. If you include a file specification that does not have a file type, the LIBRARY command uses the default file type of LIS (LST if /RSX11 is specified.)

If you specify /LIST in conjunction with qualifiers that perform additional operations on the library, the LIBRARY command creates the listing after completing all the other requests; thus the listing reflects the status of the library after all changes have been made.

When you specify /LIST, the LIBRARY command provides, by default, the following information about the library:

DIRECTORY OF FILE file-spec

File name, file type, and version number of the library being listed.

library-type LIBRARY CREATED BY: LIB vvvvvv

The type of library (OBJECT or MACRO) and the version number of the librarian that created the library. (If you specify /RSX11, the name of the librarian is LBR.)

LAST INSERT OCCURRED dd-mmm-yyyy AT hh:mm:ss

The date and time at which the last insertion was made.

MNT ENTRIES ALLOCATED: nn; AVAILABLE: mm

The current status of the module name table: the maximum number of entries that can be entered in the table (nn) and the number of entries that are unused (mm).

GST ENTRIES ALLOCATED: nn; AVAILABLE: mm

The current status of the global symbol table: the maximum number of entries that can be entered in the table (nn), and the number of entries that are unused (mm). For a macro library, both values are always 0. (If you specify /RSX11, this line lists the entry point table (EST) entries.)

FILE SPACE AVAILABLE: nnnnn WORDS

The amount of space available in the library for new files.

```
RECOVERABLE DELETED SPACE:  nnnnn WORDS
```

The amount of space occupied by modules whose entries have been deleted from the module name table. To recover this space, use the /COMPRESS qualifier.

```
module
module
   .
   .
   .
```

The names of all the entries in the module name table.

If you specify /LIST, you can also specify /FULL and /NAMES to request additional information in the listing.

/MACRO

Indicates that the library is a macro library. When you specify /MACRO and you do not specify /RSX11, the file type of the library defaults to MLB and the file types of any input files default to MAR.

If you specify /MACRO in conjunction with /RSX11, the input file type defaults to MAC.

/NAMES
/NONAMES

Controls, when /LIST is specified for an object module library, whether the LIBRARY command lists the names of all global symbols in the global symbol table as well as the module names in the module name table.

/NONAMES is the default; if you specify /NAMES, each module entry name is displayed in the format:

```
** MODULE:entry-name

  symbol    symbol    symbol    symbol    symbol    symbol
    .         .         .         .         .         .
```

If the library is a macro library and you specify /NAMES, no symbol names are displayed.

/OBJECT

Indicates that the library is an object module library. This is the default.

/OUTPUT=file-spec

Specifies, when the /EXTRACT or /COMPRESS qualifiers are specified, the file specification of the output file.

For /EXTRACT, the output file contains the modules extracted from a library; for /COMPRESS, the output file contains the compressed library.

/REMOVE=symbol,...

Requests the LIBRARY command to delete entries for one or more global symbols from the global symbol table.

When you specify /REMOVE, the LIBRARY command displays the message:

```
GLOBAL SYMBOLS DELETED:
```

Then, it displays the names of global symbols it successfully deleted.

/REPLACE

Requests the LIBRARY command to replace one or more existing modules in a library with the modules in the input file or files specified. The LIBRARY command first deletes an existing entry, if any, for each module or macro in the input file(s) and the corresponding global symbols, then inserts the new module or macro in the library.

This is the default operation; if you specify an input file parameter and do not specify /CREATE, /INSERT, or /REPLACE, the LIBRARY command replaces an existing module(s) in the file with the modules in the files specified.

When you use the /REPLACE function, the LIBRARY command displays the names of each module replaced, in the format:

```
MODULE "module-name" REPLACED
```

/RSX11

Indicates that the specified library is a library containing object modules in RSX-11M format, or macros for the MACRO-11 Assembler.

/SELECTIVE_SEARCH

Defines the input files being inserted into a library as candidates for selective searches by the linker. If you specify /SELECTIVE_SEARCH, the modules are selectively searched by the linker when the library is specified as a linker input file: only the global symbols in the module(s) that are referenced by other modules are included in the symbol table of the output image file.

/SQUEEZE

Requests that the LIBRARY command compress individual macros before adding them to a macro library. When you specify /SQUEEZE, trailing blanks, trailing tabs, and comments are deleted from each macro before insertion in the library.

Use /SQUEEZE in conjunction with the /CREATE, /INSERT, and /REPLACE qualifiers to conserve space in a macro library.

Examples

1.
```
$ LIBRARY/CREATE TESTLIB ERRMSG,STARTUP
```

   The LIBRARY command creates an object module library named TESTLIB.OLB and places the modules ERRMSG.OBJ and STARTUP.OBJ in the library.

2.
```
$ LIBRARY/INSERT TESTLIB SCANLINE
$ LINK TERMTEST,TESTLIB/LIBRARY
```

   The LIBRARY command adds the module SCANLINE.OBJ to the LIBRARY TESTLIB.OLB. The library is specified as input to the linker by using the /LIBRARY qualifier on the LINK command. If the module TERMTEST.OBJ refers to any routines or global symbols not defined in TERMTEST, the linker will search the global symbol table of library TESTLIB.OLB to resolve the symbols.

3.
```
$ FORTRAN SCANLINE
$ LIBRARY TESTLIB SCANLINE

MODULE "SCANLINE" REPLACED
```

The FORTRAN command compiles a source program named SCANLINE.FOR and creates an object module SCANLINE.OBJ. The LIBRARY command in this example uses the default function, /REPLACE, to replace the module SCANLINE in the library TESTLIB.OLB with the new version.

4.
```
$ LIBRARY/DELETE=STARTUP TESTLIB
MODULES DELETED:

STARTUP

$ LIBRARY/LIST/NAMES TESTLIB

DIRECTORY OF FILE TESTIB.OLB;2
OBJECT MODULE LIBRARY CREATED BY LIB:  VX129.0
LAST INSERT OCCURRED 10-JUN-78 AT 16:41:23
MNT ENTRIES ALLOCATED:  75 AVAILABLE:  73
GST ENTRIES ALLOCATED:  275;  AVAILABLE:  270
FILE SPACE AVAILABLE:  18347 WORDS

RECOVERABLE DELETED SPACE:  00134 WORDS

** MODULE:ERRMSG

  ERRMSG     ERR$ERROR     ERR$FATAL     ERR$WARNING


** MODULE:SCANLINE

  SCANLINE

$ LIBRARY/COMPRESS=(BLOCKS:50)  TESTLIB
$ PURGE TESTLIB.OLB
```

The LIBRARY command deletes the module STARTUP from the library TESTLIB.OLB. The next LIBRARY command requests a listing of the contents of the library TESTLIB.OLB. The /NAMES qualifier requests a list of each of the global symbols in the modules.

The listing indicates that the deletion of the module STARTUP resulted in unused space; the /COMPRESS function deletes the space and requests that 50 blocks be allocated for the library. By default, /COMPRESS creates a new version of the library. The PURGE command purges the earlier version.

5.
```
$ LIBRARY/EXTRACT=(DESCRIPTOR,RDTERM,WRTERM) -
$__/OUTPUT=TEMP -
$__DBB2:[GOODWIN.LIB]LOCALMAC.MLB
```

The /EXTRACT qualifier names three macros, DESCRIPTOR, RDTERM, and WRTERM, to be copied from the macro library LOCALMAC.MLB in the subdirectory [GOODWIN.LIB] on the disk DBB2. The /OUTPUT qualifier requests that the output file have a file name of TEMP. The LIBRARY command writes the output to the file TEMP.MAR in your current default disk and directory.

6.
```
$ LIBRARY/MACRO/CREATE=(BLOCKS:40,MODULES:100) -
$_MYMAC TEMP
$ MACRO MYMAC/LIBRARY+CYGNUS/OBJECT
```

The LIBRARY command creates a macro library named MYMAC.MLB from the macros in the file TEMP.MAR. The new library has room for 100 modules in a 40-block file. Although only a single input file is specified, the file contains three macros. Each macro is entered in the new library.

The MACRO command assembles the source file CYGNUS.MAR; the /LIBRARY qualifier specifies the library MYMAC.MLB as an input file. If the source file CYGNUS contains any macro calls not defined within the file, the assembler searches the library.

7.
```
$ LIBRARY/LIST=MYMAC.LIS/FULL MYMAC.MLB
```

The LIBRARY command requests a full listing of the macro library MYMAC; the output is written to a file named MYMAC.LIS.

LINK

Invokes the VAX-11 Linker to link one or more object modules into a program image and defines execution characteristics of the image. This command is described in detail in the VAX-11 Linker Reference Manual.

If you use the /RSX11 qualifier, the LINK command invokes the RSX-11M Task Builder. The LINK/RSX11 command is described under its own heading.

Format

```
LINK    file-spec,...

  Command Qualifiers                    Default

  /BRIEF
  /[NO]CONTIGUOUS                       /NOCONTIGUOUS
  /[NO]CROSS_REFERENCE                  /NOCROSS_REFERENCE
  /[NO]DEBUG[=file-spec]                /NODEBUG
  /[NO]EXECUTABLE[=file-spec]           /EXECUTABLE
  /FULL
  /[NO]MAP[=file-spec]                  /NOMAP
  /[NO]SHAREABLE[=file-spec]            /NOSHAREABLE
  /[NO]SYMBOL_TABLE[=file-spec]         /NOSYMBOL_TABLE
  /[NO]SYSLIB                           /SYSLIB
  /[NO]SYSSHR                           /SYSSHR
  /[NO]SYSTEM[=base-address]            /NOSYSTEM
  /[NO]TRACEBACK                        /TRACEBACK


  File Qualifiers

  /INCLUDE=module-name,...
  /LIBRARY
  /OPTIONS
  /SELECTIVE_SEARCH
```

Prompts

```
File:       file-spec,...
```

Command Parameters

file-spec,...

Specifies one or more input files. The input files can be object modules to be linked, libraries to be searched for external references or from which specific modules are to be included, shareable images to be included in the output image, or option files to be read by the linker. If you specify multiple input files, separate the file specifications with commas (,) or plus signs (+). In either case, the linker creates a single image file.

If you do not specify a file type in an input file specification, the linker supplies default file types, based on the nature of the file. All object modules are assumed to have file types of OBJ.

Command Qualifiers

/BRIEF

Requests the linker to produce a brief map (memory allocation) file. /BRIEF is valid only if /MAP is also specified.

A brief form of the map contains:

- A summary of the image characteristics
- A list of all object modules included in the image
- A summary of link-time performance statistics

/CONTIGUOUS
/NOCONTIGUOUS

Controls whether the output image file is contiguous. By default, the image file is not contiguous.

/CROSS_REFERENCE
/NOCROSS_REFERENCE

Controls whether the memory allocation listing (map) contains a symbol cross reference. /CROSS_REFERENCE is valid only if /MAP is also specified and /BRIEF is not specified; /CROSS_REFERENCE must follow /MAP in the command line.

A symbol cross reference lists each global symbol referenced in the image, its value, and all modules in the image that refer to it.

/DEBUG[=file-spec]
/NODEBUG

Controls whether a debugger is included in the output image.

If the object module contains local symbol table and/or traceback information for the debugger, you can specify /DEBUG to include the information in the image as well. If the object module does not contain symbol table and/or traceback information, and you specify /DEBUG, only global symbols are available for symbolic debugging.

The /DEBUG qualifier optionally accepts the name of an alternate, user-specified debugger. If a file specification is entered, and it does not contain a file type, the linker assumes the default file type of OBJ. If you specify /DEBUG without a file specification, the default VAX/VMS Symbolic Debugger is linked with the image. For information on using the debugger, see the VAX-11 Symbolic Debugger Reference Manual.

/EXECUTABLE[=file-spec]
/NOEXECUTABLE

Controls whether the linker creates an executable image and optionally provides a file specification for the output image file.

By default, the linker creates an executable image with the same file name as the first input file and a file type of EXE. When you specify /EXECUTABLE, you can control the defaults applied to the output file specification by the placement of the qualifier in the command, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers."

Use /NOEXECUTABLE to determine the outcome of linking a set of modules, without incurring the linker overhead required to create an image file.

/FULL

Requests the linker to produce a full map (memory allocation) listing. /FULL is valid only if /MAP is specified.

A full listing contains the following information:

- All the information included in the brief listing
- Detailed descriptions of each program section and image section in the image file
- Lists of global symbols by name and by value

/MAP[=file-spec]
/NOMAP

Controls whether a memory allocation listing (map) is produced and optionally defines the file specification. If you specify /MAP, you can also specify /BRIEF, /FULL or /CROSS_REFERENCE to control the contents of the map. If you do not specify any of these qualifiers, the map contains:

- All the information contained in a brief listing
- A list of user-defined global symbols by name
- A list of user-defined program sections

When you specify /MAP, you can control the defaults applied to the output file specification, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers."

/SHAREABLE[=file-spec]
/NOSHAREABLE

Requests the linker to produce a shareable image file rather than an executable image.

Shareable images cannot be run with the RUN command. However, they can be linked with object modules to create executable images. By default, the linker creates an executable image. If you specify both of the qualifiers /EXECUTABLE and /SHAREABLE, the /SHAREABLE qualifier always takes precedence.

When you specify /SHAREABLE, you can control the defaults applied to the output file specification by the placement of the qualifier in the command, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers."

/SYMBOL_TABLE[=file-spec]
/NOSYMBOL_TABLE

Requests the linker to create a separate file containing symbol definitions for all global symbols in the image. The output file will be in object module format.

If you also specify /DEBUG, the linker includes the global symbol definitions in the image for use by the debugger, and also creates a separate symbol table file.

The symbol table file can be used as input to subsequent LINK commands, to provide the symbol definitions to other images.

By default, the linker does not create a symbol table file.

When you specify /SYMBOL_TABLE, you can control the defaults applied to the output file specification, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers." The output file type always defaults to STB.

/SYSLIB
/NOSYSLIB

Controls whether the default shareable image and system library are to be automatically searched for unresolved references.

By default, the linker searches the default system shareable image SYS$LIBRARY:VMSRTL.EXE and then the default system library SYS$LIBRARY:STARLET.OLB when it cannot resolve references using the input file(s) specified in the command.

If you specify /NOSYSLIB, neither VMSRTL.EXE nor STARLET.OLB is searched.

/SYSSHR
/NOSYSSHR

Controls whether the linker searches the default system shareable image when it cannot resolve references in the input file(s) specified.

By default, the linker searches the default system shareable image SYS$LIBRARY:VMSRTL.EXE and then the default system library SYS$LIBRARY:STARLET.OLB when it cannot resolve references using the input file(s) specified. Use the /NOSYSSHR qualifier to request that only the default system library be searched.

/SYSTEM[=base-address]
/NOSYSTEM

Requests the linker to produce a system image and optionally defines a base address for the image. A system image cannot be run with the RUN command; it must be bootstrapped or otherwise loaded into memory.

The base address specifies the virtual memory location at which the image is to be loaded. The address can be expressed in decimal, hexadecimal, or octal, using the radix specifiers %D, %X, or %O, respectively. If you do not specify a base address, the linker uses the default address of %X80000000.

If you specify /SYSTEM, you cannot specify either /SHAREABLE or /DEBUG.

System images are intended for special purposes, such as standalone operating system diagnostics. When the linker creates a system image, it orders the program sections in alphanumeric order and ignores all program section attributes.

/TRACEBACK
/NOTRACEBACK

Controls whether the linker includes traceback information in the image file. By default, the linker includes traceback information so that the system can trace the call stack when an error occurs. If you specify /NOTRACEBACK, no traceback reporting when an error occurs.

If you specify /DEBUG, /TRACEBACK is assumed.

File Qualifiers

/INCLUDE=module-name,...
Indicates that the associated input file is an object module library, and that only the module names specified are to be unconditionally included as input to the linker.

At least one module name must be specified. If you specify more than one module name, separate them with commas and enclose the list in parentheses.

If you specify /INCLUDE, you can also specify /LIBRARY; then, the library is subsequently searched for unresolved references.

/LIBRARY
Indicates that the associated input file is a library to be searched for modules to resolve any undefined symbols in the input files.

If the associated input file specification does not include a file type, the linker assumes the default file type of OLB. You cannot specify a library as the first input file unless you also specify the /INCLUDE qualifier to indicate which modules in the library are to be included in the input. You can use both /INCLUDE and /LIBRARY to qualify a file specification. In this case, the explicit inclusion of modules occurs first, then the library is used to search for unresolved references.

/OPTIONS
Indicates that the associated input file contains a list of options to control the linking. If you specify /OPTIONS and the associated input file specification does not include a file type, the linker uses the default file type of OPT.

For complete details on the contents of an options file, see the VAX-11 Linker Reference Manual.

/SELECTIVE_SEARCH
Indicates that the associated input file is an object module, and that any symbols defined in the module that are not necessary to resolve outstanding references should be excluded from the symbol table of the output image file and also from the symbol table file, if /SYMBOL_TABLE is specified. The binary code in the object module is always included.

Examples

1. `$ LINK ORION`

   The linker links the object module in the file ORION.OBJ and creates an executable image named ORION.EXE.

2. `$ LINK/MAP/FULL DRACO,CYGNUS,LYRA`

   The linker links the modules DRACO.OBJ, CYGNUS.OBJ, and LYRA.OBJ and creates an executable image named DRACO.EXE. The /MAP and /FULL qualifiers request a full map of the image, with descriptions of each program section, lists of global symbols by name and by value, and a summary of the image characteristics. The map file is named DRACO.MAP.

3.
```
$ LINK [SSTEST]SERVICE/INCLUDE=DRACO, -
$_[]CYGNUS/EXECUTABLE
```

The LINK command links the object module DRACO from the library SERVICE.OLB in the directory SSTEST with the module CYGNUS.OBJ in the current default directory. The executable image is named CYGNUS.EXE. The placement of the /EXECUTABLE qualifier provides the output file name default.

4.
```
$ LINK/MAP/CROSS_REFERENCE/EXECUTE=DBGWEATH -
$_/DEBUG -
$_WEATHER,MATHLIB/LIBRARY
$ RUN DBGWEATH

        DEBUG, Version 1.0

%DEBUG-I-INITIAL, language is FORTRAN, scope and module set
to 'WEATHER'

DBG>
```

The linker links the object module WEATHER.OBJ with the debugger. If any unresolved references are encountered, the linker searches the library MATHLIB.OLB before searching the system library. The /CROSS_REFERENCE qualifier requests a cross reference listing in the map file; the map file is named, by default, WEATHER.MAP. The /EXECUTE qualifier requests the linker to name the output file DBGWEATH.EXE. The RUN command executes the image; the message from the debugger indicates that it is ready to accept debug commands.

LINK/RSX11

Invokes the RSX-11M Task Builder to build an RSX-11M image.

Format

```
LINK/RSX11   file-spec,...
```

Command Qualifiers	Default
/BRIEF	/BRIEF
/[NO]DEBUG[=file-spec]	/NODEBUG
/DEFAULT_LIBRARY=file-spec	
/[NO]EXECUTABLE[=file-spec]	/EXECUTABLE
/[NO]EXIT[=n]	/NOEXIT
/FULL	/BRIEF
/[NO]HEADER	/HEADER
/[NO]MAP[=file-spec]	/NOMAP
/[NO]OVERLAY_DESCRIPTION	/NOOVERLAY_DESCRIPTION
/[NO]POSITION_INDEPENDENT	/NOPOSITION_INDEPENDENT
/[NO]POST_MORTEM	/NOPOST_MORTEM
/[NO]SEQUENTIAL	/NOSEQUENTIAL
/[NO]SYMBOL_TABLE	/NOSYMBOL_TABLE
/TKB_OPTIONS=file-spec	
/[NO]TRACE	/NOTRACE
File Qualifiers	
[NO]CONCATENATED	/CONCATENATED
/INCLUDE=module,...	
/LIBRARY	
/SELECTIVE_SEARCH	

Prompts

```
File:      file-spec,...
```

Command Parameters

file-spec,...

File specifications of one or more input files. The input files may be object modules to be linked or libraries to be searched for external references. If multiple input files are specified, they may be separated either with commas (,) or plus signs (+). In either case, a single RSX-11M image file is produced.

If a file specification does not contain a file type, the task builder supplies default file types, based on the nature of the file. All object modules are assumed to have a file type of OBJ; libraries are assumed to have a file type of OLB; overlay descriptor files are assumed to have a file type of ODL.

Command Qualifiers

/BRIEF

Requests the task builder to produce a brief map (memory allocation) file; /BRIEF is the default if /MAP is specified. The /BRIEF qualifier is valid only if /MAP is also specified. A brief form of the map contains:

- A summary of the image attributes
- A list of all segments in the image
- A summary of task builder statistics

/DEBUG[=file-spec]
/NODEBUG

Controls whether or not an image is bound with a debugger. The /DEBUG qualifier optionally accepts the name of an alternate, user-specified debugging aid. If a file specification is entered, and it does not contain a file type, the task builder assumes the default file type of OBJ.

If you specify /DEBUG without a file specification, the default debugger, ODT, is used.

/DEFAULT_LIBRARY=file-spec

Defines an object module library to use in place of the default system library, SYSLIB.OLB. The specified library is searched after all libraries specified as input files when unresolved references are encountered.

/EXECUTABLE[=file-spec]
/NOEXECUTABLE

Controls whether or not the task builder produces an executable image and optionally provides a file specification for the output file.

By default, the task builder creates an image with the same file name as the first input file and a file type of EXE. If the first input file you specify is the name of a library qualified with /INCLUDE, then the default file name for the object module created is the same as the name of the first, or only module specified with /INCLUDE.

Use /NOEXECUTABLE when you want to determine the outcome of task building a set of modules without incurring the task builder overhead required to create an image.

When you specify /EXECUTABLE, you can control the defaults applied to the output file specification by the placement of the qualifier in the command, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers." The output file type always defaults to EXE.

/EXIT[=n]
/NOEXIT

Controls whether the task builder exits after a specified number of error diagnostics. By default, the task builder does not exit because of diagnostic errors. If you specify /EXIT, the task builder exits after n diagnostic errors (n is assumed to be a decimal number, by default). If you specify /EXIT and do not specify a value for n, it defaults to a value of 1.

/FULL

Requests the task builder to produce a full map (memory allocation) listing. The /FULL qualifier is valid only if /MAP is specified. A full map contains the following information:

- All the information included in the brief map
- A file contents section for each module in the image
- A list of global symbol definitions by module
- A list of unresolved global symbol references

/HEADER
/NOHEADER

Controls whether the task builder includes a task header in the image and in the symbol table file.

/MAP[=file-spec]
/NOMAP

Controls whether or not a memory allocation listing (map) is produced and optionally defines the file specification. If /MAP is specified, the qualifiers /BRIEF or /FULL can also be specified to control the contents of the map. If neither of these qualifiers is specified, /BRIEF is the default.

When you specify /MAP, you can control the defaults applied to the output file specification by the placement of the qualifier in the command, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers." The output file type always defaults to MAP.

/OVERLAY_DESCRIPTION
/NOOVERLAY_DESCRIPTION

Indicates whether the input file describes an overlay structure for the image. If the input file specification does not contain a file type, the task builder uses the default file type of ODL. If you specify /OVERLAY_DESCRIPTION, you can specify only a single input file; the input file must contain the input file specifications and an overlay description.

/POSITION_INDEPENDENT
/NOPOSITION_INDEPENDENT

Indicates whether or not the image being built contains position independent code. By default, the task builder assumes that code is not position independent.

/POST_MORTEM
/NOPOST_MORTEM

Controls whether the task builder sets the Post Mortem Dump flag. If you specify /POST_MORTEM, the system automatically lists the contents of memory when the image terminates abnormally.

/SEQUENTIAL
/NOSEQUENTIAL

Controls whether the task builder reorders program sections alphabetically when it creates the image. If you specify /SEQUENTIAL, the task builder orders program sections in the order in which they are input.

/SYMBOL_TABLE[=file-spec]
/NOSYMBOL_TABLE

Requests the task builder to create a separate file in object module format containing symbol definitions for all symbols contained in the image.

If /DEBUG is specified, the task builder includes the symbol definitions in the image for use by the debugger, and also creates a separate symbol table file.

The symbol table file can be used as input to subsequent LINK/RSX11 commands, to provide the symbol definitions to other images.

By default, the task builder does not create a symbol table file. When you specify /SYMBOL_TABLE, you can control the defaults applied to the output file specification by the placement of the qualifier in the command, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers." The output file type always defaults to STB.

/TKB_OPTIONS=file-spec

Specifies the name of a file containing task builder options. If the file specification does not include a file type, the default file type of CMD is assumed. You must omit the initial slash character (/) in an options file specified as input to the LINK/RSX11 command.

/TRACE
/NOTRACE

Indicates whether the image is traceable. If you specify /TRACE, a trace trap occurs following the execution of each instruction when the image is executed.

File Qualifiers

/CONCATENATED
/NOCONCATENATED

Indicates whether the associated input file consists of concatenated object modules. By default, the task builder includes in the image all the modules in the file. If you specify /NOCONCATENATED, the task builder includes only the first module in the file.

/INCLUDE=module-name,...

Indicates that the associated input file is an object module library, and that only the module names specified are to be unconditionally included as input to the task builder.

At least one module name must be specified. If you specify more than one module name, separate them with commas and enclose the list in parentheses.

If you specify /INCLUDE, you cannot specify /LIBRARY; if you want the library to also be searched for unresolved references, you must specify the library file specification a second time.

/LIBRARY
Indicates that the associated input file is an object module library that is to be searched for modules that resolve any undefined symbols in the input files.

If the associated input file specification does not include a file type, the task builder assumes the default file type of OLB.

You cannot specify a library as the first input file.

/SELECTIVE_SEARCH
Indicates that the associated input file is an object module, and that any symbols defined in the module that are not necessary to resolve outstanding references should be excluded from the symbol table of the output image file and also from the symbol table file, if /SYMBOL_TABLE is specified. The binary code in the object module is always included.

Examples

1.
```
$ LINK/RSX11 AVERAGE
$ RUN AVERAGE
```

The object module AVERAGE.OBJ is linked to create the task image named AVERAGE.EXE. The RUN command executes the task.

2.
```
$ LINK/RSX11 WEATHER,MATHLIB -
$_/INCLUDE=(TEMP,PRECIP), -
$_MATHLIB/LIBRARY -
$_/DEBUG/EXECUTABLE=DBGWEATH
```

The task builder links the object module WEATHER.OBJ with the modules TEMP and PRECIP from the library MATHLIB.OLB; MATHLIB is respecified as an input file with the /LIBRARY qualifier to indicate that MATHLIB should also be searched for any unresolved references.

The image includes the system debugging aid, ODT. The output RSX-11M image file is named DBGWEATH.EXE.

3.
```
$ COBOL/RSX11 PAYROLL
$ RUN SYS$SYSTEM:MRG
PLEASE ENTER FILE SPECIFICATION FOR OUTPUT FILE
PAYROL.ODL
    .
    .
$ LINK/RSX11 PAYROL/OVERLAY
```

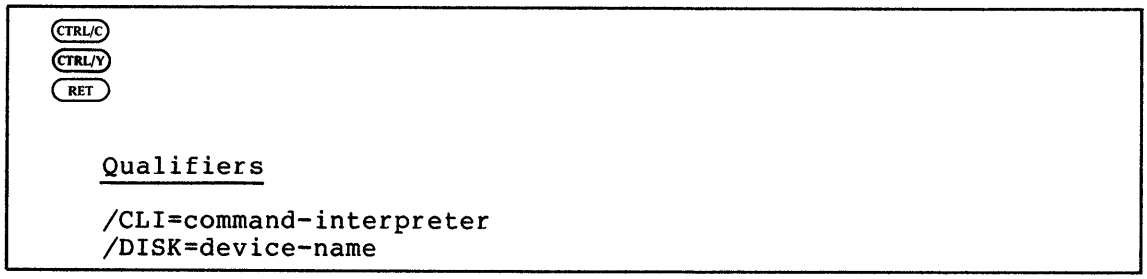
The COBOL/RSX11 command invokes the COBOL compiler to compile a source program named PAYROLL.CBL. The LINK/RSX11 command specifies the name of the overlay description file, PAYROL.ODL, created by the MERGE program.

Login Procedure

There is no LOGIN command. Rather, you get the attention of the system and signal your intention to access the system by pressing CTRL/C, CTRL/Y, or carriage return on a terminal not currently in use. The system then prompts you for your user name and your password, and then validates them.

Specify qualifiers following the entry of your user name.

Format



Prompts

Username: user-name

Password: password

Description

The login procedure:

- Validates your right to access the system by checking your user name and password against the entries in the user authorization file.
- Establishes the default characteristics of your terminal session based on your user name entry in the authorization file.
- Executes either the command procedure file named LOGIN.COM if one exists in your default directory or the command file defined in user authorization file, if any.

Qualifiers

/CLI=command-interpreter
Specifies the name of an alternate command interpreter to override the default command interpreter listed in the user authorization file.

/DISK=device-name
Specifies the name of a disk device to be associated with the logical device SYS$DISK for the terminal session. This specification overrides the default SYS$DISK device established in the authorization file.

Examples

1. (CTRL/Y)

```
Username: HUMPTY
Password:
```

CTRL/Y gets the attention of the operating system, which immediately prompts for user name. After validating the user name, the system prompts for the password, but does not echo it.

2. (RET)

```
Username: HIGGINS/DISK=DBB2
Password:

        VAX/VMS Version 1.0

$ SHOW DEFAULT
  DBB2:[HIGGINS]
```

The /DISK qualifier requests that the default disk for the terminal session be DBB2. The SHOW DEFAULT command response shows that DBB2 is the default disk.

LOGOUT

Terminates an interactive terminal session.

Format

```
LOGOUT

  Command Qualifiers

  /BRIEF
  /FULL
```

Prompts

None.

Description

You must use the LOGOUT command to end a terminal session. If you turn the power off at your terminal without using the LOGOUT command from a direct line, you remain logged in.

Command Qualifiers

/BRIEF
Requests the brief form of the logout message. The command interpreter displays your user name and the date and time at which you logged out. /BRIEF is the default for an interactive job.

/FULL
Requests the long form of the logout message. When you specify /FULL, the command interpreter displays a summary of accounting information for the terminal session. /FULL is the default for a batch job.

Examples

1.
```
$ LOGOUT
   HIGGINS   logged out at 23-JAN-1978 17:48:56.73
```

2.
```
$ LOGOUT/FULL
   HIGGINS   logged out at 24-JAN-1978 14:23:45.30

Accounting information:
Buffered I/O count:           22     Peak working set size:    90
Direct I/O count:             10     Peak virtual size:        69
Page faults:                  68     Mounted volumes:           0
Elapsed CPU time:     0 00:01:30.50  Elapsed time:     0 04:59:02.6
```

The LOGOUT command displays a summary of accounting statistics for the terminal session.

MACRO

Invokes the VAX-11 MACRO assembler to assemble one or more assembly language source programs.

If you specify /RSX11, the MACRO command invokes the MACRO-11 assembler; all the other qualifiers apply to both the VAX-11 and the MACRO-11 assembler.

For more information on the VAX-11 MACRO assembler, see the VAX-11 MACRO User's Guide. For information on the MACRO-11 assembler, see the RSX-11M MACRO User's Guide.

Format

```
MACRO       file-spec,...

    Command Qualifiers          Default

    /RSX11

    File Qualifiers

    /DISABLE=function,...       /DISABLE=(AMA,DBG,FPT,LSB)
    /ENABLE=function,...        /ENABLE=(GBL,TBK)
    /LIBRARY
    /[NO]LIST[=file-spec]
    /[NO]OBJECT[=file-spec]
    /[NO]SHOW[=function,...]    /SHOW=(CND,MC,MD)
```

Prompts

```
File:       file-spec,...
```

Command Parameters

file-spec,...

Specifies one or more assembly language source files to be assembled. If you do not specify a file type for an input file, the assembler uses the default file type of MAR. If you specify /RSX11, the MACRO-11 assembler uses the default file type of MAC.

You can specify more than one input file. If you separate the file specifications with commas (,), each file is assembled separately. If you separate the file specifications with plus signs (+), the files are concatenated and assembled as a single input file, producing single object and listing files.

Command Qualifiers

/RSX11

Indicates that the assembly language source statements are MACRO-11 source statements, and that the MACRO-11 assembler should be invoked.

File Qualifiers

/DISABLE=function,...
/ENABLE=function,...

Provides initial settings for the functions controlled by the assembler directives. .ENABL and .DSABL. You must specify at least one of the functions listed below. You can use either the short form (equivalent to the assembler directive syntax) or the long form. You can enable or disable:

AMA	ABSOLUTE	Assembly of relative addresses as absolute addresses
DBG	DEBUG	Inclusion of local symbol table information in the object file for use with the debugger
FPT	TRUNCATION	Truncation of floating-point numbers (if truncation is disabled, numbers are rounded)
GBL	GLOBAL	Assumption that undefined symbols in the assembly are external symbols
LSB	LOCAL_BLOCK	Local symbol blocks to cross PSECT boundaries, or to contain nonlocal symbols
TBK	TRACEBACK	Providing information to the debugger traceback mechanism

/LIBRARY

Indicates that the associated input file is a macro library. If you do not specify a file type, the assembler uses the default file type of MLB.

If you specify more than one macro library as input files, the libraries are searched in reverse order of their specification when a macro call is issued in a source program. There is no forward searching; when a macro library contains definitions for macro calls in a source program, you must specify the library before you specify the name of the source program.

/LIST[=file-spec]
/NOLIST

Controls whether an output listing is created, and optionally provides an output file specification for the listing file.

If you issue the MACRO command interactively, the assembler, by default, does not create a listing file. When /NOLIST is present, either explicitly or by default, errors are reported on the current output device.

If you execute the MACRO command in a batch job, /LIST is the default. When you specify /LIST, you can control the defaults applied to the output file specification by the placement of the qualifier in the command, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers." The default file type provided for listing files is LIS (LST if /RSX11 is specified).

/OBJECT[=file-spec]
/NOOBJECT

Controls whether an object module is created by the assembler. It also defines the file specification for the file.

By default, the assembler creates an object module with the same file name as the first input file and a file type of OBJ. When you specify /OBJECT, you can control the defaults applied to the output file specification by the placement of the qualifier in the command, as described in Section 6.3.3, "Rules for Entering Output File Qualifiers." The default file type provided for object files is OBJ.

/SHOW[=function,...]
/NOSHOW[=function,...]

Provides initial settings for the functions controlled by the assembler directives .LIST and .NLIST. You can specify one or more of the functions listed below. You can use either the short form (equivalent to the assembler directive syntax) or the long form. If you specify /SHOW without any functions, the listing level count is incremented. If you specify /NOSHOW without any functions, the listing level count is decremented.

CND	CONDITIONALS	List unsatisfied conditional code associated with .IF and .ENDC directives
MC	CALLS	List macro calls and repeat range expansions
MD	DEFINITIONS	List macro definitions
ME	EXPANSIONS	List macro expansions
MEB	BINARY	List binary code generated by the expansion of macro calls.

For additional details on these functions, see the VAX-11 MACRO Language Reference Manual.

Examples

1. `$ MACRO ORION`

   The MACRO assembler assembles the file ORION.MAR and creates an object file named ORION.OBJ. If this command is executed in a batch job, the assembler also creates a listing file named ORION.LIS.

2. `$ MACRO/LIST CYGNUS, LYRA/OBJECT=LYRAN`

   This MACRO command requests two separate assemblies. The MACRO command assembles CYGNUS.MAR to produce CYGNUS.LIS and CYGNUS.OBJ. Then it assembles LYRA.MAR and creates a listing file named LYRA.LIS and an object module named LYRAN.OBJ.

3.
```
$ MACRO  MYLIB/LIBRARY + ALPHA/LIST/OBJECT-
$_ + [TEST]OLDLIB/LIBRARY + []BETA
$ PRINT ALPHA
```

The MACRO command concatenates the files ALPHA.MAR and BETA.MAR to produce an object file named ALPHA.OBJ and a listing file named ALPHA.LIS. MYLIB.MLB (in the current default directory) and OLDLIB.MLB (in the directory TEST) are specified as libraries to be searched for macro definitions. When macro calls are found in BETA.MAR, OLDLIB, MYLIB, and the system library STARLET.MLB are searched, in that order, for the definitions. When macro calls are found in ALPHA.MAR, MYLIB.MLB and STARLET.MLB are searched; OLDLIB is not searched.

The PRINT command prints the listing.

MCR

Provides a means of running RSX-11M components in a manner that is compatible with the RSX-11M operating system.

Format

```
MCR   [component[command-string]]

   Command Qualifiers

   None.
```

Prompts

MCR>

Command Parameters

component
: Is the RSX-11M command used to invoke the desired component, for example, MAC or TKB. If you do not specify a component, MCR prompts for a command.

  The component name must be equivalent to the file name portion of the file specification for the component as it exists on the VAX/VMS system disk, that is, SYS$SYSTEM:filename. For example, you must type MAC, not MACRO, to invoke the MACRO-11 assembler.

command-string
: Is a valid command string for the component. If you do not specify a command string, the requested component prompts for a command string.

Description

For information on how to use the MCR command interpreter under VAX/VMS and for a list of the RSX-11M components available under VAX/VMS, see the VAX-11/RSX-11M User's Guide.

Examples

1.
```
$ MCR DSP MYFILE.DAT
$
```

The MCR command precedes a single RSX-11M command. When the command finishes, DCL prompts for another command.

2.
```
$ MCR
MCR>PIP MYFILE.DAT/SP
MCR>^Z
$
```

The MCR command requests activation of MCR command mode. The MCR> prompt indicates that the MCR command interpreter is ready to accept commands. After the PIP command executes, MCR continues prompting until you use CTRL/Z to return to DCL.

3.
```
$ MCR PIP
PIP> MYFILE.DAT/SP
PIP> ^Y
```

The MCR command requests PIP, which in turn prompts for a command string. Pressing CTRL/Y returns control to DCL.

MOUNT

Makes a volume and the files or data it contains available for processing by system commands or user programs.

Format

```
MOUNT   device-name,...   [volume-label,...]   [logical-name[:]]

   Command Qualifiers[1]          Default

   /DATA_CHECK[=option,...]
   /FOREIGN
   /OVERRIDE=option,...
   /OWNER_UIC=uic
   /PROCESSOR=option
   /PROTECTION=code
   /[NO]WRITE                     /WRITE

   Qualifiers for Disks           Default

   /ACCESSED=n
   /EXTENSION=n
   /GROUP
   /[NO]SHARE                     /NOSHARE
   /SYSTEM
   /UNLOCK
   /WINDOWS=n

   Qualifiers for Tapes           Default

   /BLOCK=n                       /BLOCK=2048
   /DENSITY=n
   /[NO]LABEL                     /LABEL
   /RECORD=n
```

Prompts

Device: device-name

Label: volume-label

Log_Name: logical-name

Command Parameters

device-name,...

Specifies the physical device name or logical name of the device on which the volume is to be mounted.

If you specify more than one device name for a multivolume tape file, separate the device names with either commas (,) or plus signs (+); you can specify more volume labels than device names.

---

1 For convenience qualifiers that are applicable only to disks and to tapes are listed separately. All qualifier descriptions appear in alphabetical order, however.

volume-label,...
Specifies the label on the volume. For disk volumes, the label can have up to 12 characters; for tape volumes, the label can have up to 6 characters.

If you specify more than one volume label for a multivolume tape set, separate them with either commas (,) or plus signs (+). The volumes must be in the same volume set and the labels must be specified in the correct order.

The volume-label parameter is not required when you mount a volume with the /FOREIGN qualifier, nor when you specify /OVERRIDE=IDENTIFICATION. To specify a logical name when you enter either of these qualifiers, type any alphanumeric characters in the volume label parameter position.

logical-name
Defines a 1- to 63-alphanumeric character string logical name to be associated with the device.

If you do not specify a logical name, the MOUNT command assigns the default logical names DISK$volume-label and TAPE$volume-label to disk and tape devices, respectively.

The MOUNT command places the name in the process logical name table, unless you specify /GROUP or /SYSTEM. In the latter cases, it places the logical names in the group or system logical name tables.

Description

When you issue the MOUNT command, the MOUNT command checks:

- That a volume is physically loaded on the device specified
- That the label on the volume matches the label specified
- Your UIC against the owner UIC on the volume to ensure that you have the right to access the volume.

For additional information and examples of using the MOUNT command, see Chapter 3, "Disk and Tape Volumes."

Command Qualifiers

/ACCESSED=n
Specifies, for disk volumes, the approximate number of directories that will be in use, concurrently, on the volume. You can specify a value from 0 through 255 to override the default specified when the volume was initialized.

The user privilege OPER is required to use /ACCESSED.

/BLOCK=n
Specifies, for tape volumes, the default block size.

Legal values are in the range of 18 through 65534. By default, records are written to tape volumes in 2048-byte blocks. For foreign or unlabeled tapes, the default is 512-bytes.

/DATA_CHECK[=options,...]
Overrides the read-check and/or write-check options specified for a volume when it was initialized. You can specify either or both of the following options:

READ	Perform checks following all read operations
WRITE	Perform checks following all write operations

If you specify /DATA_CHECK without specifying an option, the system assumes /DATA_CHECK=WRITE.

/DENSITY=n
Specifies, for foreign or unlabeled tapes, the density (in bpi) at which the tape will be written. You can specify either 800 or 1600. If you do not specify a density for a tape that was previously written, the density defaults to that at which the tape was written.

The specified density is used only if you specify /FOREIGN or /NOLABEL and the first operation performed on the tape is a write.

If you specify /LABEL, or if the first operation on the tape is a read, the tape is read or written at the density at which the first record on the tape is recorded.

/EXTENSION=n
Specifies, for disk files, the number of blocks by which files will be extended on the volume unless otherwise specified by an individual command or program request.

You can specify a value from 0 through 65535 to override the value specified when the volume was initialized.

/FOREIGN
Indicates that the tape or disk volume is not in the standard ANSI format used by the VAX/VMS operating system.

If you mount a volume with the /FOREIGN qualifier, the program you use to read the volume must be able to process the labels on the volume, if any.

For example, DOS volumes must be mounted with the /FOREIGN qualifier and processed with the File Transfer Utility (FLX).[1]

The default protection applied to foreign volumes is RWLP for the system and owner. If /GROUP is also specified, group members are also given RWLP (Read, Write, Logical I/O and Physical I/O) access. If /SYSTEM or /SHARE is specified, the group and world are both given RWLP access. If you mount a volume currently in Files-11 format with the /FOREIGN qualifier, you must have override the volume protection privilege (VOLPRO), or you must be the owner of the volume.

---

[1] FLX is an RSX-11M utility program; for information on how to use RSX-11M utilities under VAX/VMS, see the VAX-11/RSX-11M User's Guide.

/GROUP

Makes the volume available to other users with the same group number in their user identification codes (UICs) as the user issuing the MOUNT command.

The logical name for the device is placed in the group logical name table. You must have the privilege to place a name in the group logical name table (GRPNAM) to use the /GROUP qualifier.

/LABEL
/NOLABEL

Indicates, for tape volumes, whether the tape contains standard labels.

If you mount a tape with the /NOLABEL qualifier, an end-of-file condition is returned when a tape mark is encountered during reading the tape.

The default protection for unlabeled tapes is all access to the system and owner, and no access to the group and world.

/OVERRIDE=option,...

Inhibits one or more of the following protection checks that the MOUNT command performs:

ACCESSIBILITY (For tapes only). Allows you to override a nonblank VOL1 or HDR1 accessibility field. You must have the user privilege to override volume protection (VOLPRO) or be the owner of the volume.

EXPIRATION (For tapes only). Allows you to write a tape that has not yet reached its expiration date. You must have the user privilege to override volume protection (VOLPRO) or be the owner of the volume.

IDENTIFICATION Allows you to mount a volume when you do not know what the volume label is. If you specify /OVERRIDE=IDENTIFICATION, you can specify anything for the volume-label parameter or you can omit it; the MOUNT command ignores whatever you enter. The volume must be mounted /NOSHARE (either explicitly or by default).

SETID (For tapes only). Allows you to inhibit checks of the volume set identification when you switch reels in a multi-volume tape set.

/OWNER_UIC=uic

Requests that the specified user identification code be assigned ownership of the volume while it is mounted, overriding the ownership recorded on the volume. Or, if you are mounting a volume /FOREIGN, requests an owner UIC other than your current UIC.

Specify the UIC in the format:

[g,m]

g is an octal number in range of 0 through 377 representing the group number.

m is an octal number in the range of 0 through 377 representing the member number.

The brackets are required.

You must have the user privilege to override volume protection to use this qualifier for a Files-11 volume; or, you must be the owner of the volume.

/PROCESSOR=option

Requests that the MOUNT command associate an ancillary control program (ACP) to process the volume, overriding the default manner in which ACPs are associated with devices. The specified option can be one of the following:

UNIQUE	Create a new process to execute a copy of the default ACP image for the specified device type or controller.
SAME:device	Use the same ACP process currently being used by the device specified.
file-spec	Create a new process to execute the specified ACP image (for example, a modified or a user-written ACP).

You must have operator privilege to use the /PROCESSOR qualifier.

/PROTECTION=code

Specifies the protection code to be assigned to the volume.

Specify the code according to the standard syntax rules for specifying protection. (These rules are given in the discussion of the SET PROTECTION command.) If you omit a protection category, that category of user is denied all access.

If you do not specify a protection code, the protection defaults to that assigned to the volume when it was initialized. If you specify /PROTECTION when you mount a volume with the /SYSTEM or /GROUP qualifiers, the protection code specified overrides any access rights implied by the other qualifiers.

If you specify /FOREIGN, the Execute and Delete access codes have no meaning; you can however, specify the access codes P (Physical I/O) and/or L (Logical I/O) to restrict the nature of input/output operations different user categories can perform.

You must have the user privilege to override volume protection to use the /PROTECTION qualifier on a Files-11 volume, or you must be the owner of the volume.

/RECORD=n

Specifies, for tape volumes, the number of characters in each record.

The record size must be less than or equal to the block size specified or used by default.

Use /RECORD in conjunction with /BLOCK to de-block records.

/SHARE
/NOSHARE
Indicates, for a disk volume, whether the volume is shareable. If the volume has already been mounted by another user, and you request that it be mounted with the /SHARE qualifier, any other qualifiers you enter are ignored.

By default, the MOUNT command assumes that a volume is not shareable, and allocates the device on which it is mounted.

If you have previously allocated the device and specify the /SHARE qualifier, the MOUNT command deallocates the device so that other users can access it.

/SYSTEM
Makes the volume available to all users in the system, as long as the UIC-based volume protection allows them access.

The logical name for the device is placed in the system logical name table. You must have the privilege to place a name in the system logical name table (SYSNAM) to use the /SYSTEM qualifier.

/UNLOCK
Requests, for disk volumes, write access to the index file on the volume.

The /UNLOCK qualifier is valid only if the volume is mounted /NOSHARE.

/WINDOWS=n
Specifies the number of mapping pointers to be allocated for file windows. When a file is opened, the file system uses the mapping pointers to access data in the file.

You can specify a value from 7 to 80 to override the default value specified when the volume was initialized.

You must have operator privilege to use the /WINDOWS qualifier.

/WRITE
/NOWRITE
Controls whether the volume can be written.

By default, a volume is considered read/write when it is mounted. You can specify /NOWRITE to provide read-only access to protect files. It is equivalent to write-locking the device.

Examples

1.
```
$ MOUNT   MT:  -
$_MATH06  STAT_TAPE
%MOUNT-I-MOUNTED, MATH06 mounted on _MTA0:
$ COPY   ST061178.DAT   STAT_TAPE:
```

The MOUNT command requests that the magnetic tape whose volume label is MATH06 be mounted on the device MTA0 and assigns the logical name STAT_TAPE to the volume.

Subsequently the COPY command copies the disk file ST061178.DAT to the tape.

2.
```
$ ALLOCATE DM:
_DMB2:  ALLOCATED
$MOUNT DMB2:  TEST_FILES
%MOUNT-I-MOUNTED, TEST_FILES mounted on _DMB2:
```

The ALLOCATE command requests an available RK06 device. After noting the device name in the response from the ALLOCATE command, the physical volume can be placed on the device. Then, the MOUNT command mounts the volume.

3.
```
$ MOUNT/SHARE DMA2:  PUB_FILES -
$_/PROTECTION=(SYS:RWED,OWN:RWED,GR:RWED)
```

The MOUNT command mounts the volume PUB_FILES as a shareable volume. The /PROTECTION qualifier allows all types of access to the system, the owner, and to other members of the volume owner's group. All other users are denied all types of access.

4.
```
$ MOUNT/FOREIGN   MTA1:
%MOUNT-I-MOUNTED, TESTER mounted on _MTA1:
$ MCR FLX
```

The MOUNT command requests the mounting of a foreign volume on the device MTA1. The MCR FLX command invokes the RSX-11M File Transfer Utility to process the volume.

5.
```
$ MOUNT   DMA1:   PUBS_BACK
%SYSTEM-F-INCVOLLABEL, incorrect volume label
-MOUNT-I-VOLIDENT, label='BACK_UP_GMB', owner='MALCOLM',
 format= 'DECFILE11B'
```

The system response to the first MOUNT command indicates that the label was incorrectly specified. The second message from the MOUNT command displays the label on the volume, the volume owner, and the volume format (DECFILE11B indicates that the volume is in DIGITAL's Files-11, Structure Level 2 format).

ON

Defines the default courses of action when a command or program executed within a command procedure (1) encounters an error condition or (2) is interrupted by CTRL/Y. The specified actions are taken only if the command interpreter is enabled for error checking or CTRL/Y interrupts; these are the default conditions.

Formats

```
ON   severity-level   THEN   [$] command

   Command Qualifiers            Default

   None.                         ON ERROR THEN $ EXIT

ON   CONTROL_Y   THEN   [$] command
```

Prompts

None.

Command Parameters

severity-level
Specifies the severity condition that will cause the action indicated to be taken. The severity-level is represented by one of the following keywords:

```
WARNING
ERROR
SEVERE_ERROR
```

You can truncate any of these keywords to one or more characters.

command
Specifies the action to be taken when errors equal to or greater than the specified level of error occur. You can specify any valid command line after the keyword THEN; you can optionally precede the command line with a dollar sign.

Description

Error Handling: When any command or program is executed, the return status value is checked with the current ON severity level. If you specify WARNING, the specified action is taken for warning, error, and severe error returns. If you specify ERROR, the specified action is taken for error and severe error returns; the default action for warnings is to continue. If you specify SEVERE_ERROR, the specified action is taken only for severe, or fatal, errors; the default action for errors and warnings is to continue.

If you do not include an ON statement in a command procedure, the command interpreter, by default, issues the EXIT command whenever errors or severe errors occur during command or program execution. In all other cases, command execution continues.

Any ON statement in a command procedure overrides the effect of a previous ON statement. The effect of an ON statement is canceled whenever the specified command is executed as a result of the specified error condition. Then, the default action, to stop on errors and severe errors, is reinstated.

The action specified by an ON command applies only within the command procedure in which the command is executed. If you specify an ON command in a procedure that executes another procedure, the ON command action does not apply to the nested procedure.

You can request the command interpreter not to check the status returned from commands with the SET NOON command; restore checking with the SET ON command.

CTRL/Y Handling: Normally, pressing CTRL/Y during the execution of a command procedure causes the command interpreter to receive control. In a command procedure, you can specify an action to be taken when CTRL/Y is pressed; this action also applies when CTRL/C is pressed to interrupt a command or program that has no CTRL/C handling routine.

Any ON CONTROL_Y statement overrides the effect of a previous statement. An ON CONTROL_Y action remains in effect until explicitly overridden by another ON CONTROL_Y action statement or until the SET NOCONTROL_Y command is executed.

Note the following characteristics of CTRL/Y action handling:

- If a procedure that has set a CTRL/Y action invokes another procedure, the default action for CTRL/Y for the new command level is to exit. If CTRL/Y is pressed while this level is active, control returns to the procedure that specified the CTRL/Y action; execution continues following the @ command.

- If an image is executing when CTRL/Y is pressed, exit handlers declared by the image, if any, are executed before the CTRL/Y action is taken.

- If no image is currently executing, that is, the command interpreter is executing a command that does not invoke an image, the command completes before the CTRL/Y action is taken.

- If the CTRL/Y action does not result in a transfer of control in the procedure, then the procedure continues executing following the CTRL/Y action. Execution resumes following the command or program image that was interrupted.

You can disable CTRL/Y interrupts with the SET NOCONTROL_Y command. Then, neither a default nor a user-specified action is taken when CTRL/Y is pressed. Note that a SET NONCONTROL_Y command followed by a SET CONTROL_Y command resets the default CTRL/Y condition handling.

Examples

1.
```
$ ON SEVERE_ERROR THEN CONTINUE
```

After this statement is executed in a command procedure, execution of the procedure will continue when any warning, error, or severe error condition occurs. Once the statement has been executed as a result of a fatal error condition, the default action, EXIT, is reinstated.

2.
```
$ ON ERROR THEN GOTO BYPASS
$ RUN A
$ RUN B
   .
   .
$ EXIT
$ BYPASS:  RUN C
```

If either program A or program B returns a status code with a severity level of error or severe error, control is transferred to the statement labeled BYPASS.

3.
```
$ ON WARNING THEN EXIT
   .
   .
$ SET NOON
$ RUN [SSTEST]LIBRA
$ SET ON
   .
   .
```

The ON command requests that the procedure exit when any warnings, errors, or severe errors occur. Later, the SET NOON command disables error checking before executing the RUN command. Regardless of any status code returned by the program LIBRA.EXE, the procedure continues. The next command, SET ON, re-enables error checking and reestablishes the initial ON condition.

4.
```
$ ON CONTROL_Y THEN GOTO CTRL_EXIT
   .
   .
$ CTRL_EXIT:
$ CLOSE INFILE
$ CLOSE OUTFILE
$ EXIT
```

The ON command specifies action to be taken when CTRL/Y is pressed during the execution of this procedure. When CTRL/Y is pressed, the GOTO command that transfers control to the line labeled CTRL_EXIT is executed. At this label, the procedure performs clean-up operations, in this example, closing files and exits.

OPEN

Opens a file for reading or writing at the command level.

Format

```
OPEN   logical-name  file-spec

   Command Qualifiers        Default

   /ERROR=label
   /READ                     /READ
   /WRITE                    /READ
```

Prompts

Log_Name: logical-name

File: file-spec

Command Parameters

logical-name
Specifies a logical name to be assigned to the file.

file-spec
Specifies the name of the file or device to be opened for input or output. If the file specification does not include a file type, the system uses the default file type of DAT.

Description

A file can be opened for either reading or writing, but not both. After a file is opened, it is available for input or output at the command level with the READ and WRITE commands, or it can be accessed within a program image for input or output operations using the logical name.

The logical devices SYS$INPUT, SYS$OUTPUT, SYS$COMMAND, and SYS$ERROR do not have to be explicitly opened before they can be read or written at the command level. All other files must be explicitly opened.

You can issue more than one OPEN command for the same file, and assign it different logical names. If you specify the same logical name on more than one OPEN command without closing the file, no warning message is issued and the file is not opened. If the file had been previously read, the next read request returns the record after the record previously read.

Command Qualifiers

/ERROR=label
Specifies a label on a line in the command procedure to receive control if the open request results in an error. If no error routine is specified and an error occurs during the opening of the file, the command procedure continues execution at the next line in the file, as it does if no error occurs.

The error routine specified for this qualifier takes precedence over any action statement indicated in an ON command. If /ERROR is not specified, the current ON condition action is taken.

If an error occurs and a target label is successfully given control, the reserved global symbol $STATUS contains a successful completion status.

/READ

Requests that the file be opened for reading. This is the default, if neither /READ nor /WRITE is specified.

/WRITE

Requests that the file be opened for writing. If the file specification does not contain a version number and the file already exists, a new version of the file is created.

Examples

1.
```
$ OPEN INPUT_FILE AVERAGE.DAT
$ READ_LOOP:
$ READ/END_OF_FILE=ENDIT  INPUT_FILE  NUM
    .
    .
$ GOTO READ_LOOP
$ ENDIT:
$ CLOSE INPUT_FILE
```

The OPEN command opens the file named AVERAGE.DAT as an input file and assigns it the logical name INPUT_FILE. The READ command reads a record from the logical file INPUT_FILE into the symbol named NUM. The procedure executes the lines between the labels READ_LOOP and ENDIT until the end of the file is reached. At the end of the file, the CLOSE command closes the file.

2.
```
$ OPEN/WRITE/ERROR=OPEN_ERROR  OUTPUT_FILE  TEMP.OUT
$ WRITE_LOOP:
    .
    .
$ GOTO WRITE_LOOP
    .
    .
$ OPEN_ERROR:
$ WRITE SYS$OUTPUT "Cannot open file TEMP.OUT"
$ EXIT
```

The OPEN command opens the file TEMP.OUT for output and assigns it the logical name OUTPUT_FILE. The /ERROR qualifier specifies that if any error occurs while opening the file, the command interpreter should transfer control to the line at the label OPEN_ERROR.

PASSWORD

Specifies the password associated with the user name specified on a JOB card for a batch job submitted through the system card reader.

Format

```
PASSWORD   password

   Command Qualifiers

   None.
```

Prompts

None.

Command Parameters

password
Specifies the 1- to 8-character password associated with the user name specified on the JOB command.

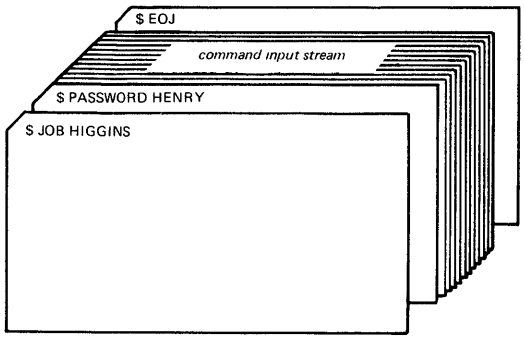
Description

The PASSWORD command is valid only in a batch job submitted through the card reader. The $ is required. The password is checked against the password associated with user name specified on the JOB card, and must be correct or the job is rejected.

Note that you can suppress printing on the card when you keypunch the PASSWORD card to prohibit other users from seeing the password.

Example

1.



The JOB and PASSWORD commands precede a batch job submitted from the card reader. An EOJ command marks the end of the job.

PRINT

Queues one or more files for printing, either on a default system printer or on a specified device.

Format

```
PRINT   file-spec,...

   Command Qualifiers          Default

   /AFTER=absolute-time
   /DEVICE=device-name
   /FORMS=type
   /HOLD                       /NOHOLD
   /[NO]IDENTIFY               /IDENTIFY
   /JOB_COUNT=n                /JOB_COUNT=1
   /LOWERCASE                  /NOLOWERCASE
   /NAME=job-name
   /PRIORITY=n
   /QUEUE=queue-name


   File Qualifiers

   /BURST                      /NOBURST
   /COPIES=n                   /COPIES=1
   /DELETE                     /NODELETE
   /[NO]FEED                   /FEED
   /FLAG_PAGE                  /NOFLAG_PAGE
   /HEADER                     /NOHEADER
   /PAGE_COUNT=n
   /SPACE[=n]                  /SPACE=1
```

Prompts

File: file-spec,...

Command Parameters

file-spec,...

Specifies one or more files to be printed. If you specify more than one file, separate the file specifications with either commas (,) or plus signs (+). In either case, the PRINT command concatenates the files into a single print job.

You can specify the file name, type, or version fields as wild cards (*).

If you do not specify a file type for the first input file, the PRINT command uses the default file type of LIS.

Note that the PRINT command cannot print a file that resides on a device that is allocated.

Description

A file or files queued by the PRINT command are considered a job. The system assigns a unique job identification (jobid) to each job in the system and displays this jobid when the PRINT command completes execution.

Printer queues are identified by name; if you do not specify a queue name with the /DEVICE or /QUEUE qualifiers, the system queues the job to an available queue. If no physical printer is currently available with required characteristics, or if the job is in a hold status, the job is placed in the queue named SYS$PRINT. The PRINT command, by default, displays the name of the queue on which it entered the job.

Command Qualifiers

/AFTER=absolute-time
Requests that the file not be printed until a specific time of day.

Specify the time using the standard syntax rules for specifying absolute time values.

If the specified time has already passed, the file is queued for printing immediately.

/DEVICE=device-name
Requests that the file(s) specified be queued for printing on a specific device to which queueing is allowed. If /DEVICE is not specified, files are queued, by default, to SYS$PRINT. This qualifier is synonymous with the /QUEUE qualifier.

/FORMS=type
Specifies the forms type required for the specified file(s).

Specify the forms type using a numeric value or alphanumeric code. Codes for forms types are installation-defined.

/HOLD
/NOHOLD
Controls whether the file is available for printing immediately. When you specify /HOLD, the file is not released for actual printing until you use the SET QUEUE command to release it.

/IDENTIFY
/NOIDENTIFY
Controls whether the PRINT command displays a message indicating the jobid of the print job and the name of the queue in which it is entered.

By default, the PRINT command displays this information whenever a job is successfully queued.

/JOB_COUNT=n
Requests that the entire job be printed n times, where n is a decimal number from 1 to 255.

/LOWERCASE
/NOLOWERCASE
Indicates that the specified file(s) contain lowercase alphabetic letters and must be printed on a printer that prints uppercase and lowercase letters.

/NAME=job-name
Defines a 1- to 8-alphanumeric character name string to identify the job. The job-name is displayed by the SHOW QUEUE command and is printed in the top and bottom rows of the flag page for the job.

If you do not specify /NAME, the name string defaults to the file name (truncated to eight characters, if necessary) of the first, or only, file in the job.

/PRIORITY=n
Specifies the priority of the print job. The priority, n, must be in the range of 0 through 31, where 0 is the lowest priority and 31 is the highest.

By default, jobs are assigned the same priority as your current process priority. The user privilege ALTPRI is required to set a priority value that is higher than your current process's priority.

/QUEUE=queue-name
Requests that the specified file(s) be printed on a specific device. This qualifier is synonymous with the /DEVICE qualifier.

File Qualifiers

/BURST
/NOBURST
Controls whether a burst page is included on output. A burst page precedes a flag page and contains the same information. However, it is printed over the perforation between pages.

Use this qualifier to override the installation-defined defaults set up for printers when they are started.

/COPIES=n
Specifies the number of copies to print. By default, the PRINT command prints a single copy of a file; you can use /COPIES to request up to 255 copies.

If you specify /COPIES after the PRINT command name, each file in the parameter list is printed the specified number of times.

If you specify /COPIES following a file specification, only that file is printed the specified number of times.

/DELETE
/NODELETE
Controls whether files are deleted after printing. If you specify /DELETE after the PRINT command name, all files specified are deleted. If you specify /DELETE after a file specification, only that file is deleted after it is printed.

The protection applied to the file must allow delete access to the current UIC.

/FEED
/NOFEED
Controls whether the PRINT command automatically inserts form feeds when it prints files that do not have carriage control characters. By default, the PRINT command inserts a form feed when the forms are within four lines of the end of the form. On standard 66-line forms, a form feed occurs after 62 lines are printed.

/FLAG_PAGE
/NOFLAG_PAGE
Controls whether a flag page is printed preceding output.

If you specify /FLAG with the command name, a flag page is printed for the entire job.

If you specify /FLAG with any file specification, a separate flag page is printed preceding the associated file.

/HEADER
/NOHEADER
Controls whether the name of the file is printed at the top of each output page. By default, the file specification is printed only at the top of the first page of output.

/PAGE_COUNT=n
Specifies the number of pages of the specified file to print. You can only use /PAGE_COUNT to qualify file specifications; it cannot qualify the command name.

/SPACE[=n]
Specifies the number of spaces (that is, blank lines) to leave between lines of output on the specified file(s). You can specify 1 or 2 to indicate the number of blank lines.

If you do not specify /SPACE, no extra lines are printed between lines of output; if you specify /SPACE without a value for n, output is double-spaced.

Examples

1.
```
$ PRINT AVERAGE
  Job 236 entered on queue LPA1
```

The PRINT command queues the file AVERAGE.LIS for printing on a system printer. The system displays the jobid and the name of the printer to which it queued the file.

2.
```
$ PRINT  ALPHA.TXT+BETA+GAMMA
  Job 237 entered on queue LPA1:
```

The PRINT command prints the files ALPHA.TXT, BETA.TXT, and GAMMA.TXT as a single file. The printer output file is named ALPHA.TXT.

3.
```
$ ASSIGN  LPA0:  SYS$PRINT
$ PRINT *.TXT
  Job 238 entered on queue LPA0:
```

The ASSIGN command creates an entry in the process logical name table for the logical name SYS$PRINT to override the system logical name definition. Subsequently, a PRINT command translates the logical name SYS$PRINT and queues the job on the queue LPA0. This job consists of the highest versions of all files with the file type of TXT.

4.
```
$ PRINT/COPIES=10/AFTER=20   ALPHA.TXT
  Job 237 entered on queue LPA1:
```

The PRINT command queues 10 copies of the file ALPHA.TXT to the printer, but requests that the copies not be printed until after 8:00 P.M.

5.
```
$ PRINT/LOWERCASE   ALPHA.TXT/COPIES=2, -
$_BETA.DOC/COPIES=3
  Job 238 entered on queue LPA1:
```

The print job queued by this print command consists of two copies of ALPHA.TXT followed by three copies of BETA.DOC. This job must be printed on a printer that can print lowercase letters.

6.
```
$ PRINT/JOB_COUNT=3   ALPHA.TXT,BETA/NOIDENTIFY
```

This PRINT command concatenates the files ALPHA.TXT and BETA.TXT into a single print job, and prints three copies of the job. The /NOIDENTIFY qualifier requests that the jobid not be displayed.

7.
```
$ PRINT/HOLD   MASTER.DOC
  Job 240 entered on queue SYS$PRINT
    .
    .
    .
$ SET QUEUE/RELEASE/ENTRY=240
```

The PRINT command queues a copy of the file MASTER.DOC to the default printer in a hold status. Later, the SET QUEUE command releases the hold status on the file and makes it available for printing.

PURGE

Deletes all but the highest numbered version or versions of a specified file or files.

Format

```
PURGE   [file-spec,...]

   Command Qualifiers        Default

   /KEEP=n                   /KEEP=1
   /[NO]LOG                  /NOLOG
```

Prompts

None.

Command Parameters

file-spec,...
Specifies one or more files to be purged. If you specify more than one file, separate them with either commas (,) or plus signs (+). If you do not specify a file specification, the PURGE command purges all files in the current default directory.

The PURGE command does not provide file name or file type defaults; version numbers are not allowed. You can use wild cards in either the file name or type field.

Command Qualifiers

/KEEP=n
Specifies the maximum number of versions to retain of the specified file(s).

If you do not specify /KEEP, all but the most recent n version(s) of the specified file(s) are deleted.

/LOG
/NOLOG
Controls whether the PURGE command displays the file specifications of files as it deletes them.

By default, PURGE does not display the file specifications of files it purges.

Examples

1. `$ PURGE *.COM`

   The PURGE command deletes all but the most recent version of each file with a file type of COM.

2. `$ PURGE AVERAGE.FOR/KEEP=2`

   The PURGE command deletes all but the two highest numbered versions of the file AVERAGE.FOR.

3. `$ PURGE`

   The PURGE command deletes all but the most recent versions of all files in the default directory.

4. ```
   $ PURGE [MALCOLM.TESTFILES]/LOG
   %PURGE-I-FILPURGED, DBA1:[MALCOLM.TESTFIILES]AVERAGE.OBJ;1 deleted
   %PURGE-I-FILPURGED, DBA1:[MALCOLM.TESTFILES]BACKUP.OBJ;2 deleted
   ```

   The PURGE command purges all files cataloged in the subdirectory named MALCOLM.TESTFILES. The /LOG qualifier requests the PURGE command to display the specification of files it deletes.
   ```

READ

Reads a single record from a specified input file and equates the record to a specified symbol name.

Format

```
READ   logical-name  symbol-name

   Command Qualifiers

   /END_OF_FILE=label
   /ERROR=label
```

Prompts

Log_Name: logical-name

Symbol: symbol-name

Command Parameters

logical-name
: Specifies the logical name of the input file from which a record is to be read. The OPEN command assigns a logical name to a file and places the name in the process logical name table.

  If the READ command is executed interactively and the logical name is specified as SYS$INPUT, SYS$OUTPUT, SYS$COMMAND, or SYS$ERROR, the command interpreter prompts for input data.

symbol-name
: Specifies a 1-to-255 alphanumeric character symbol name to be equated to the contents of the record being read. The symbol is always defined locally to the current command level. If the symbol is already defined, the READ command redefines it to the new value.

Description

The READ command can read data only from sequential files or devices. After each record is read from the specified file, the READ command positions the record pointer at the next record in the file.

The maximum size of any record that can be read in a single READ command is 255 bytes.

Before a file can be written, it must be opened with the OPEN command and assigned a logical name. The process permanent files identified by the logical names SYS$INPUT, SYS$OUTPUT, SYS$ERROR, and SYS$COMMAND do not have to be explicitly opened to be read.

Command Qualifiers

/END-OF-FILE=label

Specifies the label on a line in the current command procedure. When the last record in the file is read, the command interpreter transfers control to the line at the specified label.

If /END_OF_FILE is not specified, then control is given to the error label specified with the /ERROR qualifier, if any, when the end-of-file is reached. If neither /ERROR nor /END is specified, then the READ command that results in an end-of-file condition returns with an error status code.

/ERROR=label

Specifies a label on a line in the command procedure to receive control if the read request results in an error. If no error routine is specified and an error occurs during the reading of the file, the command procedure continues execution at the next line in the file, as it does if no error occurs.

The error routine specified for this qualifier takes precedence over any action statement indicated in an ON command. If /ERROR is not specified, the current ON condition action is taken.

If an error occurs and the target label is successfully given control, the reserved global symbol $STATUS contains a successful completion value.

Examples

1.
```
$ OPEN IN NAMES.DAT
$ LOOP:
$ READ/END_OF_FILE=ENDIT IN NAME
    .
    .
$ GOTO LOOP
$ ENDIT:
$ CLOSE IN
```

The OPEN command opens the file NAMES.DAT for input and assigns it the logical name of IN. The READ command specifies the label ENDIT to receive control when the last record in the file has been read. The procedure loops until all records in the file have been processed.

2.
```
$ READ/ERROR=READERR/END_OF_FILE=OKAY  MSGFILE  CODE
    .
    .
$ OKAY:
$ CLOSE MSGFILE
$ EXIT
$ READERR:
$ CLOSE MSGFILE
```

The READ command specifies labels to receive control at the end-of-file and on error conditions. At the end-of-file, the procedure uses the CLOSE command to close the file and exits. When an error occurs, the procedure closes the file.

3.
```
    .
    .
$ READ SYS$COMMAND  DATA_LINE
$ WRITE  OUTPUT_FILE  DATA_LINE
    .
    .
```

The READ command requests data from the current SYS$COMMAND device. If the command procedure containing these lines is executed interactively, the command issues a prompt to the terminal, accepts a line of data, and equates the data entered to the symbol name DATA_LINE.

Then, the WRITE command writes the value of the symbol DATA_LINE to the file identified by the logical name OUTPUT_FILE.

RENAME

Changes the directory name, file name, file type, or file version of an existing disk file.

Format

RENAME input-file-spec output-file-spec	
Command Qualifiers	Default
/[NO]NEW_VERSION	/NEW_VERSION

Prompts

From: input-file-spec

To: output-file-spec

Command Parameters

input-file-spec

Specifies the names of one or more files whose specifications are to be changed.

You can specify the file name, type, or version fields of the file specification with wild cards. In this case, all files whose specifications satisfy the fields that are specified are renamed.

output-file-spec

Specifies the new file specification to be applied to the input file. The RENAME command uses the device, directory, file name, and file type of the input file specification to provide defaults for nonspecified fields in the output file. If you do not specify a version number for the output file, the RENAME command increments the highest existing version, if any, by one.

You can specify a wild card in place of the file name, type, or version number of the output-file-spec; the RENAME command uses the corresponding field in the input file specification to name the output file. Wild cards in corresponding fields of the input and output file specification result in multiple rename operations. If you are renaming multiple files and specify a wild card for the file name or file type, you must specify a wild card for the version number as well.

Command Qualifiers

/NEW_VERSION
/NONEW_VERSION

Controls whether the RENAME command automatically assigns a new version number to the output file, if a file with the same file name and file type already exists.

Examples

1. `$ RENAME AVERAGE.OBJ OLDAVE`

   The RENAME command changes the file name of the highest existing version of the file AVERAGE.OBJ to OLDAVE.OBJ. If no file named AVERAGE.OLD currently exists, the new file is given a version number of 1.

2. `$ RENAME/NONEW_VERSION SCANLINE.OBJ;2 BACKUP.OBJ;*`

   The RENAME command renames the file SCANLINE.OBJ;2 to BACKUP.OBJ;2. The /NONEW_VERSION qualifier ensures that, if BACKUP.OBJ;2 already exists, the RENAME command will not rename the file, but report the error.

3. `$ RENAME *.TXT;* *.OLD;*`

   The RENAME command renames all versions of all files with file types of TXT to have file types of OLD. The file names and version numbers are not changed.

4. `$ RENAME [HIGGINS]COMPLIB.OLB [HIGGINS.LIBRARY]`

   The RENAME command changes the directory name of the object module library COMPLIB.OLB. The library is now cataloged in the subdirectory [HIGGINS.LIBRARY].

5. `$RENAME [MALCOLM.TESTFILES]SAVE.DAT []TEST`

   The RENAME command renames the file SAVE.DAT in the directory MALCOLM.TESTFILES to TEST.DAT. The new file is cataloged in the current default directory.

REQUEST

Displays a message at a system operator's terminal, and optionally requests a reply. System operators are identified by the function(s) they perform; if more than one operator is designated for a particular function, all receive the specified message.

Format

```
REQUEST   message-text

   Command Qualifiers        Default

   /REPLY
   /TO[=operators,...]       /TO=ALL
```

Prompts

None.

Command Parameters

message-text
Specifies the text of a message to be displayed at the specified operator's terminal(s).

The message text can have a maximum of 128 characters; if you type more than one word, enclose the text in quotation marks.

Description

When you use the REQUEST command to send a message to an operator, the message is displayed at one or more operators' terminals, according to the keywords specified with the /TO qualifier. If you specify /REPLY, the message is assigned an identification number, so that the operator can respond to the message.

If you specify /REPLY, you receive the message:

```
%OPCOM-S-OPRNOTIF, operator notified, waiting...hh:mm:ss
```

When the operator responds to your request, you receive a message in the format:

```
%OPCOM-S-OPREPLY, message text entered by operator
```

If you request a reply, you cannot enter any commands until the operator responds. If you press CTRL/C, you receive the message:

```
REQUEST - Enter message or cancel with <^Z>
REQUEST - Message?
```

At this time, you can enter either a message to be displayed at the specified operator(s)' terminal(s), or you can press CTRL/Z to cancel the request. If you enter a message, that message is sent to the operator and you must continue to wait for a reply.

All messages are logged at the central operator's console and in the system operator's log file, if initialized.

Command Qualifiers

/REPLY
Requests a reply to the specified message.

If you request a reply, the message is assigned a unique identification so that the operator can respond.

/TO[=operators,...]
Specifies one or more operators to whom you wish to send the message. You can specify one or more of the following keywords. If you specify more than one keyword, separate them with commas and enclose the list in parentheses. By default, the message is sent to all terminals currently designated as operators' terminals.

ALL — Send the message to all operators.

CENTRAL — Send the message to the central system operator.

DEVICES — Send the message to operators designated to mount disk and tape volumes.

DISKS — Send the message to operators designated to mount and dismount disk volumes.

OPER1 through OPER12
Send the message to an installation-specified operator identified as OPR1, OPR2, and so on.

PRINTERS — Send the message to operators designated to respond to printer requests.

TAPES — Send the message to operators designated to mount and dismount tape volumes.

Examples

1.
```
$ PRINT/COPIES=2   REPORT.OUT/FORMS=H
  Job 401 entered on queue LPA1:
$ REQUEST/REPLY/TO=PRINTERS -
$_"Have queued job 401 as forms=H;  can you print it?"
%OPCOM-S-OPRNOTIF, operator notified, waiting...10:42:16.10
%OPCOM-S-RQSTCMPLTE, request complete
%OPCOM-S-OPREPLY, AFTER 11:00
```

The PRINT command requests that multiple copies of a file be printed using a special type of paper (/FORMS=H). After queueing the PRINT job, the REQUEST command sends a message to the system operator designated to handle users' requests about print jobs. The message asks the operator how long before the file will be printed.

The reply shows the text entered by the operator.

2.
```
$ REQUEST/REPLY  "ARE YOU THERE"
%OPCOM-S-OPRNOTIF, operator notified, waiting...14:54:30.33
^C
REQUEST-Enter message or cancel request with <^Z>
REQUEST-Message?^Z
%OPCOM-S-OPRNOTIF, operator notified, waiting... 14:59:01.38
%OPCOM-F-RQSTCAN, request was canceled
```

The REQUEST command issues a message to see if there are any operators. When no response is received, CTRL/C interrupts the request and then CTRL/Z cancels it.

RUN (Image)

Places an image into execution in the process.

You can truncate the RUN command to a single letter, R.

Format

```
RUN   file-spec

   Command Qualifiers

   /[NO]DEBUG
```

Prompts

File: file-spec

Command Parameters

file-spec
Specifies an executable image to be executed. If you do not specify a file type, the RUN command uses the default file type of EXE.

Command Qualifiers

/DEBUG
/NODEBUG
Controls, for native VAX-11 images, whether the image is to be run with the debugger. If the image was linked with the /DEBUG qualifier and you do not want the debugger to prompt, use the /NODEBUG qualifier. If the image was linked with /TRACEBACK, traceback reporting is performed when an error occurs.

If the image was not linked with the debugger, you can specify /DEBUG to request the debugger at execution time. However, if /NOTRACEBACK was specified when the image was linked, /DEBUG is invalid.

Examples

1.
```
$ RUN LIBRA
```

The image LIBRA.EXE starts executing in the process.

2.
```
$ MACRO/ENABLE=DEBUG ORION
$ LINK/DEBUG ORION
$ RUN ORION

     DEBUG Version 1.0
%DEBUG-I-INITIAL, language is MACRO, scope and module set to 'ORION
DBG>
   .
   .
$ RUN/NODEBUG ORION
```

A program is compiled, linked and run with the debugger. Subsequently, a RUN/NODEBUG command requests that the debugger, which is present in the image, not prompt. If an error occurs while the image executes, the debugger can perform traceback and report on the error.

RUN (Process)

Creates a subprocess or a detached process to execute a specified image. If you specify any of the command qualifiers listed, the RUN command creates a process.

Format

```
RUN   file-spec

   Command Qualifiers                    Default

   /[NO]ACCOUNTING                       /ACCOUNTING
   /AST_LIMIT=quota                      /AST_LIMIT=10
   /[NO]AUTHORIZE
   /BUFFER_LIMIT=QUOTA                   /BUFFER_LIMIT=10240
   /DELAY=delta-time
   /ERROR=file-spec
   /FILE_LIMIT=quota                     /FILE_LIMIT=20
   /INPUT=file-spec
   /INTERVAL=delta-time
   /IO_BUFFERED=quota                    /IO_BUFFERED=6
   /IO_DIRECT=quota                      /IO_DIRECT=6
   /MAILBOX=unit
   /MAXIMUM_WORKING_SET=quota            /MAXIMUM_WORKING_SET=200
   /OUTPUT=file-spec
   /PAGE_FILE=quota                      /PAGE_FILE=10000
   /PRIORITY=n
   /PRIVILEGES=privilege-list            /PRIVILEGES=SAME
   /PROCESS_NAME=process-name
   /QUEUE_LIMIT=quota                    /QUEUE_LIMIT=8
   /[NO]RESOURCE_WAIT                    /RESOURCE_WAIT
   /SCHEDULE=absolute-time
   /[NO]SERVICE_FAILURE                  /NOSERVICE_FAILURE
   /SUBPROCESS_LIMIT=quota               /SUBPROCESS_LIMIT=8
   /[NO]SWAPPING                         /SWAPPING
   /TIME_LIMIT=limit                     /TIME_LIMIT=0
   /UIC=uic
   /WORKING_SET=default                  /WORKING_SET=200
```

Prompts

File: file-spec

Command Parameters

file-spec

Specifies an executable image to be executed in a separate process. If the file specification does not include a file type, the RUN command uses the default file type of EXE.

Description

When you use any of the qualifiers listed above with the RUN command, the RUN command creates a process and displays the process identification (PID) of the process on SYS$OUTPUT. The process executes the image specified in the file-spec parameter. When the image completes execution, the system deletes the process.

By default, the RUN command creates a subprocess with the same UIC, current disk and directory defaults, privileges, and priority of current process, and deducts resource quotas from the current process to assign to the subprocess.

The /UIC qualifier requests the RUN command to create a detached process; you must have the user privilege DETACH to create a detached process.

Input, Output, and Error Streams: Use the following qualifiers to assign equivalence names for the logical names SYS$INPUT, SYS$OUTPUT, and SYS$ERROR for the process:

```
/INPUT
/OUTPUT
/ERROR
```

The equivalence names you specify for these process-permanent files are interpreted with the context of the process you are creating. For example, file type defaults and logical name use and translation are image- and language dependent.

Defining Process Attributes: Use the following qualifiers to override the default attributes for a process:

```
/ACCOUNTING
/PRIORITY
/PRIVILEGES
/PROCESS_NAME
/SERVICE_FAILURE
/SWAPPING
```

Assigning Resource Quotas: When you issue a RUN command to create a process, you can define quotas to restrict the amount of various system resources available to the process. The following resource quotas are deductible when you create a subprocess; that is, the values you specify are subtracted from your current quota and given to the subprocess:

Qualifier	Quota
/BUFFER_LIMIT	BYTLM
/FILE_LIMIT	FILLM
/PAGE_FILE	PGFLQUOTA
/QUEUE_LIMIT	TQELM
/SUBPROCESS_LIMIT	PRCLM

The quota amounts are returned to your current process when the subprocess is deleted.

The system defines minimum values for each specifiable quota; if you specify a quota that is below the minimum, or if you specify a deductible quota that reduces your quota below the minimum, the RUN command cannot create the process. To determine your current quotas, issue the SHOW PROCESS/QUOTAS command.

You can also specify limits for the following, non-deductible quotas:

Qualifier	Quota
/AST_LIMIT	ASTLM
/IO_BUFFERED	BIOLM
/IO_DIRECT	DIOLM
/MAXIMUM_WORKING_SET	WSQUOTA
/TIME_LIMIT	CPULM
/WORKING_SET	WSDEFAULT

Hibernation and Scheduled Wakeups: Use the following qualifiers to schedule execution of the image:

/DELAY
/INTERVAL
/SCHEDULE

If you specify any of these qualifiers, the RUN command creates the process and places it in a state of hibernation. The process cannot execute the image until it is awakened. Time values specified with these three qualifiers control when the process will be awakened to execute the specified image.

You can schedule wakeups for a specified delta time (/DELAY qualifier) or absolute time (/SCHEDULE qualifier). Optionally, you can schedule wakeups for recurrent intervals, with the /INTERVAL qualifier. If you specify an interval time, the created process is awakened to execute the specified image at fixed time intervals. When the image completes normally, the process is returned to a state of hibernation. When the next scheduled wakeup occurs, the image is reactivated. Note that if the image completes abnormally, for example, if it calls the Exit system service, it does not return to hibernation.

Use the /PROCESS_NAME qualifier to give the created process a name. You can use this process name in a subsequent STOP or CANCEL command. A STOP command terminates execution of the image in the process, and causes the process to be deleted. The CANCEL command cancels wakeup requests that are scheduled but have not yet been delivered.

Command Qualifiers

/ACCOUNTING
/NOACCOUNTING

Controls whether accounting records are to be logged for the created process. By default, all processes are logged in the system accounting file.

You must have the user privilege ACNT to disable accounting.

/AST_LIMIT=quota

Specifies the maximum number of Asynchronous System Traps (ASTs) the created process can have outstanding.

If you do not specify an AST limit quota, the default value of 10 is used; the minimum required for any process to execute is 2.

This quota is not deductible.

/AUTHORIZE
/NOAUTHORIZE

Controls, when the image to be executed is the system login image (LOGINOUT.EXE), whether login searches the user authorization file to validate a detached process.

By default, the login image checks the user authorization file whenever a detached process is created. Specify /NOAUTHORIZE to create a detached process running under the control of the command interpreter. The process permanent files specified by the /INPUT and /OUTPUT qualifiers are made available to the command interpreter for input and output.

The user privilege DETACH is required to create a detached process. Any nonspecified attributes of the created process default to the same as those of the current process.

/BUFFER_LIMIT=quota

Specifies the maximum amount of memory, in bytes, that the process may use for buffered I/O operations or temporary mailbox creation.

If you do not specify a buffered I/O quota, the default value of 10240 bytes is used; the minimum amount required for any process to execute is 1024 bytes.

This quota is deductible.

/DELAY=delta-time

Requests that the created process be placed in hibernation, and awakened after a specified time interval has elapsed.

Specify the delta-time according to the rules for delta time specifications (these rules are given in Section 6.8).

If you specify /INTERVAL with /DELAY, the first wakeup request occurs at the time specified by /DELAY and all subsequent wakeups occur at intervals as specified by /INTERVAL.

/ERROR=file-spec

Defines a 1- to 63-alphanumeric character equivalence name string for the logical device name SYS$ERROR. The logical name and equivalence name are placed in the process logical name table for the created process.

/FILE_LIMIT=quota

Specifies the maximum number of files that a process can have open at any one time.

If you do not specify an open file quota for a created process, the system uses the default value of 20. The minimum amount required for any process to execute is 2.

This quota is deductible.

/INPUT=file-spec

Defines a 1- to 63-alphanumeric character equivalence name string for the logical device name SYS$INPUT. The logical name and equivalence name are placed in the process logical name table for the created process.

/INTERVAL=delta-time
Requests that the created process be placed in hibernation and awakened at regularly scheduled intervals.

Specify the delta time specified according to the rules for entering delta times (these rules are given in Section 6.8).

If you specify /DELTA or /SCHEDULE with /INTERVAL, the first wakeup occurs at the time specified by /DELAY or /SCHEDULE, and all subsequent wakeups occur at intervals specified by /INTERVAL. If you specify neither /DELAY nor /SCHEDULE with /INTERVAL, the first wakeup occurs immediately, by default.

If the image to be executed is an RSX-11M image, the /INTERVAL qualifier has the effect of the /DELAY qualifier. Only one wakeup occurs.

/IO_BUFFERED=quota
Specifies the maximum number of system-buffered I/O operations the created process can have outstanding at a time.

If you do not specify a buffered I/O quota, the default value of 6 is used; the minimum required for any process to execute is 2.

This quota is not deductible.

/IO_DIRECT=quota
Specifies the maximum number of direct I/O operations the created process can have outstanding at a time.

If you do not specify a direct I/O quota, the default value of 6 is used; the minimum required for any process to execute is 2.

This quota is not deductible.

/MAILBOX=unit
Specifies the unit number of a mailbox to receive a termination message when the created process is deleted. If no mailbox is specified, the creating process receives no notification when the process is deleted.

Mailbox creation and use and process termination mailboxes are described in the VAX/VMS System Services Reference Manual.

/MAXIMUM_WORKING_SET=quota
Specifies the maximum size to which the image to be executed in the process can increase its working set size. (An image can increase its working set size by calling the Adjust Working Set Limit system service).

If you do not specify a working set quota, the system uses a default value of 200. The minimum value required for any process to execute is 10 pages.

This quota is not deductible.

/OUTPUT=file-spec
Defines a 1- to 63-alphanumeric character equivalence name string for the logical device name SYS$OUTPUT. The logical name and equivalence name are placed in the process logical name table for the created process.

/PAGE_FILE=quota

Specifies the maximum number of pages that can be allocated in the paging file for the process; that is, the amount of secondary storage to use during the execution of the image.

If you do not specify a paging file quota, the system uses a default value of 10000 pages. The minimum value required for a process to execute is 256 pages.

This quota is deductible.

/PRIORITY=n

Specifies the base priority at which the created process is to execute.

The priority, n, is a decimal number from 0 through 31, where 31 is the highest priority and 0 is the lowest. Normal priorities are in the range 0 through 15, and time-critical priorities are in the range of 16 through 31.

You must have the user privilege to set any process priority to set the base priority higher than your current process' priority.

If you specify a higher value when you do not have the privilege, or if you do not specify a priority, the priority defaults to the priority of the current process.

/PRIVILEGES=privilege-list

Defines user privileges for the created process. The privilege list consists of one or more of the keywords listed below. You must have a user privilege to give a process you create any privileges that you yourself do not have.

[NO]ACNT	Allow/disallow process to create processes for which no accounting messages are written
ALLPRIV	Give the process all privileges
[NO]ALLSPOOL	Allow/disallow the process to allocate spooled devices
[NO]ALTPRI	Allow/disallow the process to set priority values
[NO]BUGCHK	Allow/disallow the process to make bug check error log entries
[NO]CMEXEC	Allow/disallow the process to change its mode to executive
[NO]CMKRNL	Allow/disallow the process to change its mode to kernel
[NO]DETACH	Allow/disallow the process to create detached processes
[NO]DIAGNOSE	Allow/disallow the process to issue diagnostic I/O requests
[NO]EXQUOTA	Allow/disallow the process to exceed its quotas
[NO]GROUP	Allow/disallow the process to control other processes in the same group

[NO]GRPNAM	Allow/disallow the process to place names in the group logical name table
[NO]LOG_IO	Allow/disallow the process to issue logical I/O requests to a device
[NO]MOUNT	Allow/disallow the process to issue a mount volume QIO request
[NO]NETMBX	Allow/disallow the process to create a network device
[NO]OPER	Allow/disallow the process to perform operator functions
[NO]PHY_IO	Allow/disallow the process to issue physical I/O requests to a device
[NO]PRMCEB	Allow/disallow the process to create permanent common event flag clusters
[NO]PRMGBL	Allow/disallow the process to create permanent global sections
[NO]PRMMBX	Allow/disallow the process to create permanent mailboxes
[NO]PSWAPM	Allow/disallow the process to alter its swap mode
[NO]SAME	Allow/disallow the process to have the same privileges as the current process
[NO]SETPRV	Allow/disallow the process to give higher privileges to other processes
[NO]SYSGBL	Allow/disallow the process to create system global sections
[NO]SYSNAM	Allow/disallow the process to place names in the system logical name table
[NO]TMPMBX	Allow/disallow the process to create temporary mailboxes
[NO]VOLPRO	Allow/disallow process to override volume protection
[NO]WORLD	Allow/disallow process to control all other processes in the system

If you do not specify /PRIVILEGES, the created process has the same privileges as your current process. If you specify /PRIVILEGES=NOSAME, the created process has no privileges.

/PROCESS_NAME=process-name

Defines a 1- to 15-alphanumeric character name for the created process. The process name is implicitly qualified by the group number of the process's UIC.

If you do not specify a process name, the created process has a null name, by default.

/QUEUE_LIMIT=quota
Specifies the maximum number of timer queue entries that the created process can have outstanding at any one time. This includes timer requests and scheduled wakeup requests.

If you do not specify a timer queue entry quota, the system uses the default value of 8. A process does not require any timer queue quota in order to execute.

This quota is deductible.

/RESOURCE_WAIT
/NORESOURCE_WAIT
Enables or disables resource wait mode for the created process. By default, the system places a process in a wait state when a resource required for a particular function is not available.

If you specify /NORESOURCE_WAIT, the process will receive an error status code when a resource is unavailable. /RESOURCE_WAIT is the default.

/SCHEDULE=absolute-time
Requests that the created process be placed in hibernation and awakened at a specific time of day.

Specify the absolute time value according to the rules for entering absolute time values (these rules are given in Section 6.8).

/SERVICE_FAILURE
/NOSERVICE_FAILURE
Enables or disables system service failure exception mode for the created process. By default, if an error occurs when a process calls a system service, a status code indicating an error is returned.

If you specify /SERVICE_FAILURE, the process will encounter an exception condition if an error occurs during a system service request. /NOSERVICE_FAILURE is the default.

/SUBPROCESS_LIMIT=quota
Specifies the maximum number of subprocesses that the created process is allowed to create.

If you do not specify a subprocess limit, the system uses the default value of 8. A process does not require any subprocess quota in order to execute.

This quota is deductible.

/SWAPPING
/NOSWAPPING
Enables or disables process swap mode for the created process. By default, a process is swapped from the balance set in physical memory to allow other processes to execute.

If you specify /NOSWAPPING, the process is not swapped out of the balance set when it is in a wait state. /SWAPPING is the default. You must have the user privilege to disable process swapping to specify /NOSWAPPING for a process you create.

/TIME_LIMIT=limit

Specifies the maximum amount of CPU time allocated to the created process, in 1-millisecond units. When the time expires, the process is deleted. A CPU time limit of 0 indicates that CPU time is not restricted; this is the default.

If you restrict CPU time for a process, specify the time limit according to the rules for specifying delta time values.

/UIC=uic

Specifies that the created process is to be a detached process.

The specified uic defines a user identification code for the created process, in the format:

[g,m]

g is an octal number in range of 0 through 377 representing the group number of the process

m is an octal number in the range of 0 through 377 representing the individual member number of the process

The brackets are required.

/WORKING_SET=default

Specifies the default working set size for the created process; that is, the number of pages in the working set for the image to be executed.

If you do not specify a default working set size, the system uses the default value of 200 pages. The minimum number of pages required for a process to execute is 10 pages. The value specified cannot be greater than the working set quota (specified with the /MAXIMUM_WORKING_SET qualifier).

This quota is not deductible.

Examples

1.
```
$ RUN/PROCESS_NAME=SUBA   SCANLINE
%RUN-S-PROC_ID, identification of created process is 00133404
```

The RUN command creates a subprocess named SUBA to execute the image SCANLINE. The system gives the subprocess an identification number of 00133404.

2.
```
$ RUN/INTERVAL=1:40/PROCESS_NAME=STAT   STATCHK
%RUN-S-PROC_ID, identification of created process is 001D0123
    .
    .
$ CANCEL STAT
```

The RUN command creates a subprocess named STAT to execute the image STATCHK.EXE. The process is scheduled to execute the image at 1-hour-and-40-minute intervals. The process hibernates; however, because neither /DELAY nor /SCHEDULE is specified, the first wakeup occurs immediately.

The CANCEL command subsequently cancels the wakeup requests posted by the /INTERVAL qualifier. If the process is currently executing the image, it completes the execution and hibernates.

3.
```
$ RUN/PROCESS_NAME=LYRA  LYRA -
$_/BUFFER_LIMIT=1024 -
$_/FILE_LIMIT=4 -
$_/PAGE_FILES=256 -
$_/QUEUE_LIMIT=2 -
$_/SUBPROCESS_LIMIT=2 -
$_/OUTPUT=_TTB3: -
$_/ERROR=_TTB3:
%RUN-S-PROC_ID, identification of created process is 00121089
```

The RUN commamd creates a subprocess named LYRA to execute the image LYRA.EXE. The resource quotas assigned by command qualifiers provide the subprocess with the minimum quotas required to run. The /OUTPUT and /ERROR qualifiers assign equivalences to the logical names SYS$OUTPUT and SYS$ERROR for the subprocess. Any messages the subprocess writes to its default output devices are displayed on the terminal TTB3.

4.
```
$ RUN/UIC=[100,4]/PRIVILEGES=(SAME,NOPSWAPM) -
$_/NORESOURCE_WAIT   OVERSEER
%RUN-S-PROC_ID, identification of created process is 001D002C
```

The RUN command creates a detached process to execute under the UIC [100,4]. It executes the image OVERSEER.EXE. The RUN command gives the process all the privileges of the current process, except the ability to alter its swap mode. The /NORESOURCE_WAIT qualifier disables resource wait mode for the process.

5.
```
$ ASSIGN/GROUP  [MALCOLM.TESTFILES]   TEST
$ RUN/PROCESS=SUB  WATCH -
$_/INPUT=TEST:OUT1 -
$_/OUTPUT='F$LOGICAL("SYS$OUTPUT")
%RUN-S-PROC_ID, identification of created process is 001D002E
```

The ASSIGN command creates an entry in the group logical name table for the logical name TEST. The RUN command creates a subprocess to execute the image WATCH.EXE.

The /INPUT qualifier defines SYS$INPUT for the subprocesses. The logical name TEST defines the directory for the file OUT1.DAT. Because the logical name TEST is in the group logical name table, the logical name can be translated and referred to by the image WATCH.EXE.

The /OUTPUT qualifier uses the lexical function F$LOGICAL to translate the logical name of the current process's SYS$OUTPUT device. The equivalence name string is equated to the device SYS$OUTPUT for the subprocess.

SET

Defines or changes, for the current terminal session or batch job, characteristics associated with files and devices owned by the process.

Format:

```
SET    option

   Options

   CARD_READER
   [NO]CONTROL_Y
   DEFAULT
   MAGTAPE
   [NO]ON
   PROCESS
   PROTECTION
   QUEUE
   RMS_DEFAULT
   TERMINAL
   [NO]VERIFY
   WORKING_SET
```

Prompts

What: option

Description

The SET command options listed above are described individually in this manual. Table 2 lists all of the SET command options, including those that are generally reserved for use by system operators and managers.

Table 2
SET Command Options

Option[1]	Function
ACCOUNTING [1]	Initializes the accounting log file
CARD_READER	Defines the default ASCII translation mode for a card reader
[NO]CONTROL_Y	Disables/enables interrupts caused by CTRL/Y
DEFAULT	Establishes a device and/or directory as the current default for file specifications
DEVICE [1]	Defines device characteristics
LOGINS [1]	Allows or disallows users to log in to the system
MAGTAPE	Defines characteristics of a magnetic tape device
[NO]ON	Controls whether the command interpreter checks for an error condition following the execution of commands in a command procedure
PRINTER [1]	Defines characteristics of a printer
PROCESS	Defines execution characteristics of the current process
PROTECTION	Defines the protection status of a file or group of files, or establishes the default protection to be applied to all files subsequently created during the job
QUEUE	Changes the attributes associated with one or more entries in a printer or batch job queue
RMS_DEFAULT	Provides default multi-block and multi-buffer count values to be used by RMS for file operations
TERMINAL	Defines operational characteristics of a terminal
UIC [1]	Changes the UIC of the current process
[NO]VERIFY	Controls whether the command interpreter displays lines in command procedures as it executes them
WORKING_SET	Changes the current working set limit or quota

1 Indicates that this command is described in the VAX/VMS Operator's Guide.

SET CARD_READER

Defines the default translation mode for cards read into a system card reader. All subsequent input read into the specified card reader will be converted using the specified mode.

Format

```
SET CARD_READER    device-name

    Command Qualifiers

    /026
    /029
```

Prompts

Device: device-name

Command Parameters

device-name
Specifies the name of the card reader for which the translation mode is to be set.

The device must not be currently allocated to any other user.

Command Qualifiers

/026
Indicates that the cards were punched on an 026 punch.

/029
Indicates that the cards were punched on an 029 punch.

Example

1.
```
$ ALLOCATE CR:
  _CRA0:  ALLOCATED
$ SET CARD_READER CRA0:/029
$ COPY  CRA0:  [MALCOLM.DATAFILES]CARDS.DAT
```

The ALLOCATE command requests the allocation of a card reader by specifying the generic device name. When the ALLOCATE command displays the name of the device, the SET CARD_READER command sets the translation mode at 029. Then, the COPY command copies all the cards read into the card reader CRA0 into the file CARDS.DAT in the directory MALCOLM.DATAFILES.

SET CONTROL_Y

Controls whether the command interpreter receives control when CTRL/Y is pressed.

Format

```
SET [NO] CONTROL_Y

   Command Qualifiers

   None.
```

Prompts

None.

Description

The CTRL/Y function key provides a general-purpose escape; it can be used at any time during an interactive terminal session to interrupt the current command, command procedure, or program image.

The SET NOCONTROL_Y command is provided for use in special applications; when the SET NOCONTROL_Y command is executed in a system-specified command procedure for a particular user at login, that user can communicate only with the application program that controls the terminal.

When SET NOCONTROL_Y is in effect, the CTRL/Y function key has the same effect as a CTRL/U function followed by a carriage return.

The effect of SET NOCONTROL_Y also applies to the CTRL/C function for all commands and programs that do not have special action routines to respond to CTRL/C.

Example

1. ```
   $ SET NOCONTROL_Y
   ```

   After this command, the CTRL/Y function is disabled.
   ```

SET DEFAULT

Changes the default device and/or directory name for the current process. The new default is applied to all subsequent file specifications that do not explicitly give a device or directory name.

When you change the default device assignment, the system equates the specified device with the logical name SYS$DISK.

Format

```
SET DEFAULT    device-name

   Command Qualifiers

   None.
```

Prompts

Device: device-name

Command Parameters

device-name
Specifies a device and/or directory name to be used as the default device in file specifications.

If you specify a physical device name, terminate the device name with a colon. If you specify a directory name, you must enclose it in brackets ([ ] or < >).

The SET DEFAULT command performs logical name translation on the entire string specified, not on the left-most portion of the device name specified, as is the usual case. The translation is not recursive.

Examples

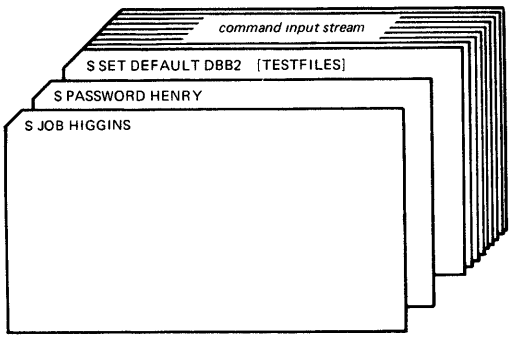
1.
```
$ SET DEFAULT [CARPENTER]
$ COPY A.* B.*
```

The SET DEFAULT command changes the default directory to CARPENTER. The default disk device does not change. The directory name CARPENTER is assumed to be the default directory for subsequent file searches, as in the COPY command shown.

2.
```
$ SET DEFAULT DBA2:
```

This command changes the default disk device to DBA2. The default directory name does not change.

3.



A batch user submits a job in the system card reader. The first command in the batch job is a SET DEFAULT command; all file specifications will default to the directory TESTFILES on the disk DBB2.

4.

```
$ SAVEDEF := 'F$DIRECTORY()
$ SET DEFAULT  [122001.MALCOLM.TESTFILES]
    .
    .
$ SET DEFAULT  'SAVEDEF'
```

This command procedure uses the F$DIRECTORY lexical function to save the current default directory in the symbol named SAVEDEF. The SET DEFAULT command changes the default directory; later the symbol SAVEDEF is used to restore the original default.

SET MAGTAPE

Defines the default characteristics associated with a specific magnetic tape device for subsequent file operations. The SET MAGTAPE command is valid for tape devices that do not currently have volumes mounted on them, or on which foreign volumes are mounted.

Format

```
SET MAGTAPE    device-name

    Command Qualifiers

    /DENSITY=density
    /REWIND
    /UNLOAD
```

Prompts

Device: device-name

Command Parameters

device-name
Specifies the name of the tape device for which the characteristics are to be set.

The device must not be currently allocated to any user.

Command Qualifiers

/DENSITY=density
Specifies the default density, in bpi (bits per inch), for all write operations on the tape device when the volume is mounted as a foreign or as an unlabeled tape. The density can be specified as either 800 or 1600.

/REWIND
Requests that the volume on the specified device be rewound to the beginning of the tape.

/UNLOAD
Requests that the volume on the specified device be rewound and unloaded.

Example

1.
```
$ MOUNT MTB1:/FOREIGN
$ SET MAGTAPE MTB1:  /DENSITY=800
```

The MOUNT command mounts a foreign tape on the device MTBl. The SET MAGTAPE command defines the density for writing the tape at 800 bpi.

SET ON

Controls whether the command interpreter performs error checking following the execution of commands in command procedures.

Format

```
SET [NO]ON

   Command Qualifiers

   None.
```

Prompts

None.

Description

During the execution of command procedures the command interpreter normally checks the status code returned when a DCL command or program image completes, and saves the numeric value of this code in the reserved symbol named $STATUS. The low-order three bits of this value are also saved in the reserved symbol $SEVERITY.

The ON command establishes an action to be taken based on the value of $SEVERITY; the default action is to exit from the current procedure when errors or severe errors are encountered.

Use the SET NOON command to override the error checking procedure. When SET NOON is in effect, the command interpreter continues to place the status code value in $STATUS and the severity level in $SEVERITY, but does not perform any action based on the value.

The SET ON or SET NOON command applies only at the current command level. If you use the SET NOON command in a command procedure that executes another procedure, the default, SET ON, is established while the second procedure executes.

Example

1.
```
$ SET NOON
$ DELETE *.SAV;*
$ SET ON
$ COPY *.OBJ *.SAV
```

This command procedure routinely copies all object modules into new files with file types of SAV. The DELETE command deletes all existing files with that file type, if any. The SET NOON command ensures that the procedure will continue execution if there are not currently any files with that file type. Following the DELETE command, the SET ON command restores error checking. Then, the COPY command makes copies of all existing files with file types of OBJ.

SET PROCESS

Changes execution characteristics associated with a process for the current terminal session or job.

Format

```
SET PROCESS    [process-name]

    Command Qualifiers

    /IDENTIFICATION=process-id
    /PRIORITY=n
    /[NO]RESOURCE_WAIT
    /[NO]SWAPPING
```

Prompts

None.

Command Parameters

process-name
Specifies the 1- to 15-alphanumeric character string name of a process whose priority is to be changed. The specified process must have the same group number in its user identification code as the current process. This parameter is invalid if the /SWAPPING or /RESOURCE_WAIT qualifiers are specified.

If you specify /IDENTIFICATION, the process-name parameter is ignored. If you specify neither the process-name parameter nor the /IDENTIFICATION parameter, the priority for your current process is changed.

Command Qualifiers

/IDENTIFICATION=process-id
Specifies the process identification the system assigned to the process when the process was created. When you specify a process identification, you can omit leading zeros.

This qualifier is invalid if you specify /RESOURCE_WAIT or /SWAPPING.

/PRIORITY=n
Specifies the new base priority for the requested process. A priority must be in the range of 0 through 31, where priorities 0 through 15 are reserved for normal processes and priorities 16 through 31 are reserved for time-critical processes.

The user privilege ALTPRI is required to increase the priority for any process to a value higher than the current process's base priority. If you do not have the ALTPRI privilege, the value you specify is compared with your current base priority and the lower value is always used.

You can use the /PRIORITY qualifier to increase or decrease a priority. Note that if you decrease your own base priority and you do not have ALTPRI privilege, you cannot restore its original value.

The GROUP and WORLD privileges are required to control other processes in the same group or other processes in the system, respectively.

/RESOURCE_WAIT
/NORESOURCE_WAIT

Enables or disables resource wait mode for the current process. By default, the system places a process in a wait state when a resource required for a particular function is not immediately available.

If you specify /NORESOURCE_WAIT, the process will receive an error status code when system dynamic memory is not available or when the process exceeds one of the following resource quotas:

Direct I/O limit
Buffered I/O limit
Buffered I/O byte count (buffer space) quota

/RESOURCE_WAIT is the default mode.

/SWAPPING
/NOSWAPPING

Enables or disables process swap mode for the current process. By default, a process that is not currently executing can be removed from physical memory so that other processes can execute.

If you specify /NOSWAPPING, the process is not swapped out of the balance set when it is in a wait state. You must have the user privilege PSWAPM to disable swapping for your process.

Examples

1.
```
$ RUN/PROCESS_NAME=TESTER CALC
%RUN-S-PROC_ID, identification of created process is 00C0012F
$ SET PROCESS TESTER/PRIORITY=10
```

The RUN command creates a subprocess and gives it the name TESTER. Subsequently, the SET PROCESS command assigns the subprocess a priority of 10.

2.
```
$ SET PROCESS/NORESOURCE_WAIT
```

The SET PROCESS command disables resource wait mode for the current process.

SET PROTECTION

Establishes the protection to be applied to a particular file or a group of files, or establishes the default protection for all files subsequently created during the terminal session or batch job. The protection for a file limits the type of access available to other system users.

Format

```
SET PROTECTION[=code]     [file-spec,...]

    Command Qualifiers

    /DEFAULT

    File Qualifiers

    /PROTECTION=code
```

Prompts

File: file-spec,...

Command Parameters

code

Defines the protection to be applied to the file(s) specified, if any; or to be applied to all files subsequently created, if /DEFAULT is specified.

The format for specifying the code is the same as that for specifying a code with the /PROTECTION qualifier. The format is described below under the heading "How to Specify Protection Codes."

file-spec,...

Specifies one or more files for which the protection is to be changed.

A file name and file type are required; if you omit a version number, the protection is changed for only the highest existing version of the file.

You can specify wild cards in the directory, file name, file type, and version fields.

If you specify /DEFAULT, you cannot enter a file specification; the specified default is applied to all files subsequently created.

Description

All disk and tape volumes have protection codes that restrict access to the volume. You can assign this protection with INITIALIZE and MOUNT commands.

For disk volumes, each file on the volume, including a directory file, can have a different protection associated with it. The SET PROTECTION command, and other file manipulating commands, allow you to define the protection for individual files.

How to Specify Protection Codes: Files can be potentially accessed by users in four categories:

- SYSTEM -- all users who have group numbers of 1 through 10 (octal). This category generally includes system managers, system programmers, and operators. It also includes all users who have either of the user privileges LOG_IO and PHY_IO.
- OWNER -- the UIC of the person who created, and therefore owns, the volume or file, and all processes that execute with that UIC.
- GROUP -- all users who have the same group number in their UICs as the owner of the file. This user category includes the owner of the file.
- WORLD -- all users who do not fall into any of the other three categories. This category includes the owner of the file and all users in the owner's group.

Each of these categories of user can be allowed or denied the following types of access:

- READ -- the right to examine, print, or copy a file or files on a volume
- WRITE -- the right to modify the file or to write files on a volume
- EXECUTE -- the right to execute files that contain executable program images (when applying protection to an entire volume, this field is interpreted as the right to create files on the volume)
- DELETE -- the right to delete the file or files on the volume

Any combination of access types can be specified for any category of user. The following syntax rules apply:

- When you specify a protection code, you must abbreviate protection types to 1 character. User categories can be entered in full or truncated to any number of characters.
- You can specify the user categories and protection types in any order.
- If you omit a protection type for a user category, that category of user is denied that type of access.
- When you specify a protection code, separate each user category from access type with a colon.

- If you specify more than one user category, separate the categories with commas and enclose the entire code in parentheses.
- When you omit a user category from a protection code applied to an entire volume, that category of user is denied all types of access.
- When you omit a user category from a protection code applied to a file or files or from a code specified for the default protection, the current access allowed that category of user remains unchanged.

For example:

```
$ SET PROTECTION=(SYS:RWED,GR:R,W)/DEFAULT
```

This protection code allows the system all types of access, group members read access only, prohibits all access by users in the world category, and does not change the current access for the owner.

Command Qualifiers

/DEFAULT
Indicates that the protection code specified is to be applied to all files subsequently created during the terminal session or job.

If you specify /DEFAULT, you cannot enter any file specifications. If you do not specify a code, the system resets the defaults established in your entry in the user authorization file.

File Qualifiers

/PROTECTION=code
Defines the protection code to be applied to the associated file specification. Use this qualifier to define different protection codes to several files in a single command.

If you specify the code parameter in addition to qualifying file specifications with the /PROTECTION qualifier, the attributes specified with the code parameter are applied first, then any attributes specified with the qualifier override them.

Specify the code in the format described above, under the heading "How to Specify Protection Codes."

Examples

1.
```
$ SET PROTECTION=(GROUP=RWED,WORLD=R)/DEFAULT
```

This SET PROTECTION command sets the default protection applied to all files subsequently created to allow other users in the same group unlimited access, and all users read access. Default protection for system and owner are not changed.

2.
```
$ SET PROTECTION PAYROLL.LIS  -
$_/PROTECTION=(SYSTEM:R,OWNER:RWED,GROUP:RW),  -
$_PAYROLL.OUT/PROTECTION=(SYSTEM:RWED,GROUP:RWED)
```

   This SET PROTECTION command changes the protection codes applied to two files. To the file PAYROLL.LIS, it gives the system read-only access, the owner read, write, execute, and delete access, and users in the owner's group read/write access. To the file PAYROLL.OUT, it gives the system and group all types of access, and does not change the default access for owner and world.

SET QUEUE

Changes the current status or attributes of a file that is queued for printing or for batch job execution but not yet processed by the system.

Format

```
SET QUEUE/ENTRY=jobid     [queue-name]

    Command Qualifiers

    /AFTER=absolute-time
    /FORMS=type
    /HOLD
    /JOB_COUNT=n
    /[NO]LOWERCASE
    /NAME=job-name
    /PRIORITY=n
    /RELEASE
```

Prompts

None.

Command Parameters

queue-name
: Specifies the name of the queue in which the specified file is entered. No logical name translation is performed on the specified queue name.

  If you do not specify a queue name, the system assumes the default name of SYS$PRINT.

Description

The system assigns a unique entry number, called a jobid, to each queued printer or batch job in the system. The PRINT and SUBMIT commands display the jobid when they successfully queue a job for processing. Use this jobid to specify the entries you want to change.

Command Qualifiers

/AFTER=absolute-time
: Requests that the specified job be held until a specific time, then released for printing. If the specified time has already passed, the file is released immediately.

  Specify the time value according to the rules for entering absolute times (these rules are given in Section 6.8).

/ENTRY=jobid
: Indicates the particular job within the queue whose characteristics are to be modified. This qualifier is required.

/FORMS=n
Modifies the forms type for the specified job. This qualifier overrides the forms type specified or defaulted on the PRINT command.

/HOLD
Requests that the specified job(s) be placed in a hold status. Jobs in a hold status are not processed until you release them with the SET QUEUE/RELEASE command.

/JOB_COUNT=n
Specifies the number of copies of the job to print. This qualifier overrides the /JOB_COUNT qualifier specified or defaulted on the PRINT command.

/LOWERCASE
/NOLOWERCASE
Indicates whether the specified job(s) must be printed on a printer with lowercase letters.

/NAME=job-name
Defines a 1- to 8-alphanumeric character name string to identify the job, overriding the job name assigned to the job when it was queued.

/PRIORITY=n
Changes the priority of a job relative to other jobs that are currently queued. The priority, n, must be in the range of 0 through 31, where 0 is the lowest priority and 31 is the highest.

By default, jobs are assigned the same priority as your current process priority; you must have the user privilege ALTPRI to set a priority value greater than your current process's priority.

/RELEASE
Releases a previously held job for processing.

Examples

1.
```
$ PRINT/HOLD   MYFILE.DAT
  Job 112 entered on queue SYS$PRINT
    .
    .
$ SET QUEUE/ENTRY=112/RELEASE/JOB_COUNT=3
```

The PRINT command requests that the file MYFILE.DAT be queued to the system printer, but placed in a hold status. The SET QUEUE command releases the file for printing and changes the number of copies of the job to three.

2.
```
$ SUBMIT  WEATHER
  Job 210 entered on queue SYS$BATCH
$ SUBMIT  CLIMATE
  Job 211 entered on queue SYS$BATCH
$ SET  QUEUE  SYS$BATCH/ENTRY=211/HOLD/NAME=TEMP
```

Two SUBMIT commands queue command procedures for batch processing. The system assigns them jobids of 210 and 211, respectively. The SET QUEUE command places the second job in a hold state and changes the job name to TEMP.

SET RMS_DEFAULT

Defines default values for the multi-block and multi-buffer counts used by VAX-11 RMS for file operations. Defaults can be set for sequential or relative files on a process-only or system-wide basis.

Format

```
SET RMS_DEFAULT

   Command Qualifiers          Default

   /BLOCK_COUNT=count
   /BUFFER_COUNT=count
   /DISK
   /INDEXED                    /SEQUENTIAL
   /MAGTAPE
   /PROCESS                    /PROCESS
   /RELATIVE                   /SEQUENTIAL
   /SEQUENTIAL                 /SEQUENTIAL
   /SYSTEM                     /PROCESS
   /UNIT_RECORD
```

Prompts

None.

Command Parameters

None.

Description

Multi-blocking and multi-buffering of file operations can enhance the speed of input/output operations with VAX-11 RMS. The defaults set with the SET RMS_DEFAULT command are applied for all file operations that do not specify explicit multi-block or multi-buffer counts.

For more information on multi-block and multi-buffer operations, see the VAX-11 Record Management Services Reference Manual.

Command Qualifiers

/BLOCK_COUNT=count

Specifies a default multi-block count for file operations. The specified count, representing the number of blocks to be allocated for each I/O buffer, can be in the range of 1 through 127.

/BUFFER_COUNT=count

Specifies a default multi-buffer count for file operations. The specified count, representing the number of buffers to be allocated, can be in the range of -128 through 127. A positive value indicates the specified number of buffers must be locked in the process's working set for the I/O operation. A negative value indicates that the specified number of buffers must be allocated but do not have to be locked.

When you use the /BUFFER_COUNT qualifier, you can use the /DISK, /INDEXED, /MAGTAPE, /RELATIVE, /SEQUENTIAL, and /UNIT_RECORD qualifiers to specify the types of file for which the default is to be applied.

/DISK
Indicates that the specified default(s) are to be applied to file operations on disk devices. If /SEQUENTIAL or /RELATIVE is specified, /DISK is assumed.

/INDEXED
Indicates that the specified default(s) are to be applied to indexed file operations. This qualifier is currently unused.

/MAGTAPE
Indicates that the specified default(s) are to be applied to operations on magnetic tape volumes.

/PROCESS
Indicates that the specified defaults are to be applied to file operations occurring within the current process.

/RELATIVE
Indicates that the specified defaults are to be applied to file operations on relative files.

/SEQUENTIAL
Indicates that the specified defaults are to be applied to all sequential file operations, including operations on disk, magnetic tape, and unit record devices.

/SEQUENTIAL is the default if neither /RELATIVE nor /INDEXED is specified.

/SYSTEM
Indicates that the specified defaults are to be applied to file operations by all processes.

The user privilege OPER is required to set the default for the system.

/UNIT_RECORD
Indicates that the specified default(s) are to be applied to file operations on unit record devices.

Examples

1.
```
$ SET RMS_DEFAULT/DISK/BLOCK_COUNT=16
```

The SET RMS_DEFAULT command defines the default multi-block count for disk file input/output operations as 16 blocks. This default is defined only for the current process, and will be used for disk file operations in user programs that do not explicitly set the multi-block count.

2.
```
$ SET RMS_DEFAULT/BUFFER_COUNT=8/MAGTAPE
```

The SET RMS_DEFAULT command defines the default multi-buffer count for input/output operations on magnetic tapes as eight buffers.

SET TERMINAL

Changes the characteristics of a specified terminal.

Format

```
SET  TERMINAL    [device-name]

   Command Qualifiers          Default

   /[NO]BROADCAST              /BROADCAST
   /CRFILL[=formula]           /CRFILL=0
   /[NO]ECHO                   /ECHO
   /[NO]EIGHT_BIT              /NOEIGHT_BIT
   /[NO]ESCAPE                 /NOESCAPE
   /[NO]HARDCOPY
   /[NO]HOLD_SCREEN            /NOHOLD_SCREEN
   /[NO]HOSTSYNC               /NOHOSTSYNC
   /[NO]INTERACTIVE            /INTERACTIVE
   /LA36
   /LFFILL[=formula]           /LFFILL=0
   /[NO]LOCAL
   /[NO]LOWERCASE
   /PAGE[=n]                   /PAGE=0
   /[NO]PARITY[=option]        /NOPARITY
   /[NO]PASSALL                /NOPASSALL
   /[NO]READSYNC               /NOREADSYNC
   /[NO]REMOTE
   /[NO]SCOPE
   /SPEED=rate
   /[NO]TAB                    /NOTAB
   /[NO]TTSYNC                 /TTSYNC
   /[NO]TYPE_AHEAD             /TYPE_AHEAD
   /UNKNOWN
   /[NO]UPPERCASE
   /VT05
   /VT52
   /VT55
   /WIDTH=n
   /[NO]WRAP                   /WRAP
```

Prompts

None.

Command Parameters

device-name

Specifies the name of the terminal whose characteristics are to be changed.

If you do not specify a device name, the qualifiers change the characteristics of the current SYS$COMMAND device, if SYS$COMMAND is a terminal.

Description

The SET TERMINAL command allows you to modify specific terminal characteristics for a particular application, or to override system default characteristics. (These defaults are defined on an individual installation basis, based on the most common type of terminal in use.)

The following qualifiers modify more than one characteristic, based on the specific type of terminal:

/LA36
/VT05
/VT52
/VT55

The settings affected by each of these qualifiers are summarized in Table 3.

Command Qualifiers

/BROADCAST
/NOBROADCAST

Controls whether the terminal can receive messages broadcast by the system operator. By default, a terminal receives any messages the system operator or another privileged user sends.

Use /NOBROADCAST when you are using a terminal as a non-interactive terminal or when you do not want special output to be interrupted by messages.

/CRFILL[=formula]

Specifies whether the system must generate fill characters following a carriage return on the terminal.

The formula is a number in the range of 0 through 9 indicating the number of null fill characters required to ensure that the carriage return completes successfully before the next meaningful character is sent. You may need to use this qualifier if you are using a non-DIGITAL terminal.

The default is /CRFILL=0.

/ECHO
/NOECHO

Controls whether the terminal echoes, or displays, the input lines that it receives.

You must have the physical I/O privilege (PHY_IO) to use the /NOECHO qualifier. When /NOECHO is set, the terminal displays only data that a system or user application program writes to it.

/EIGHT_BIT
/NOEIGHT_BIT

Indicates whether the terminal uses an 8-bit ASCII character code.

/NOEIGHT_BIT is the default; the terminal interprets characters using 7-bit ASCII code.

/ESCAPE
/NOESCAPE

Indicates whether the terminal generates valid escape sequences that will be interpreted by an applications program controlling the terminal.

If you specify /ESCAPE, the terminal checks the escape sequences for syntax before passing them to the program. For information on escape sequences, see the VAX/VMS I/O User's Guide.

/HARDCOPY
/NOHARDCOPY

Indicates whether the terminal prints hardcopy output, as opposed to a video terminal.

This qualifier is complementary to the /SCOPE qualifier, that is, /HARDCOPY is equivalent to /NOSCOPE.

/HOLD_SCREEN
/NOHOLD_SCREEN

Enables and disables the operation of the SCROLL key on VT55 video terminals.

If you specify /HOLD_SCREEN, the SET TERMINAL command also sets the /TTSYNC qualifier.

/HOSTSYNC
/NOHOSTSYNC

Controls whether the system can synchronize the flow of input from the terminal.

When you specify /HOSTSYNC, the system generates CTRL/S and CTRL/Q to enable or disable the reception of input. When the type-ahead buffer is full, the system sends CTRL/S to temporarily stop input; when the buffer is empty, the system sends CTRL/Q so that more input can be entered.

/INTERACTIVE
/NOINTERACTIVE

Indicates that the terminal is in use as an interactive terminal.

This qualifier is complementary to the /PASSALL qualifier, that is, /INTERACTIVE is equivalent to /NOPASSALL.

/LA36

Indicates that the terminal is an LA36 terminal. When you specify this qualifier, default terminal characteristics for LA36 terminals are set. These settings are summarized in Table 3.

/LFFILL[=formula]

Specifies whether the system must generate fill characters following a line feed on the terminal.

The formula is a number in the range of 0 through 9 indicating the number of null fill characters required to ensure that the line feed completes successfully before the next meaningful character is read. You may need to use this qualifier if you are using a non-DIGITAL terminal.

The default is /LFFILL=0.

/LOCAL
/NOLOCAL
Controls, for terminals attached to dial-up lines, whether the terminal can be accessed from a remote location.

This qualifier is complementary to the /REMOTE qualifier, that is, /LOCAL is equivalent to /NOREMOTE. The physical I/O privilege (PHY_IO) is required to use this qualifier.

/LOWERCASE
/NOLOWERCASE
Indicates whether the terminal has uppercase and lowercase characters.

If you specify /NOLOWERCASE all alphabetic characters are translated to uppercase.

This qualifier is complementary to the /UPPERCASE qualifier, that is, /LOWERCASE is equivalent to /NOUPPERCASE.

/PAGE=n
Specifies the page length of the terminal. For hardcopy terminals, the page size, n, equals the number of print lines between perforations on the paper. When the terminal reads a form feed character, it advances the paper to the next perforation. A page size of 0 indicates that the terminal treats form feeds as if they were line feeds.

You can specify values of 0 through 255 for the page size. The default size is installation dependent.

/PARITY[=option]
/NOPARITY
Defines the parity for the terminal. You can specify one of the following options:

EVEN
ODD

If you specify /PARITY and you do not specify an option, the command assumes /PARITY=EVEN.

/PASSALL
/NOPASSALL
Controls whether the system interprets special characters or passes all data to an application program as 8-bit binary data.

You must have the physical I/O privilege (PHY_IO) to use the /PASSALL qualifier. A terminal operating with /PASSALL set does not expand tab characters to blanks, fill carriage return or line feed characters, or recognize control characters.

/READSYNC
/NOREADSYNC
Controls whether the system solicits read data from a terminal using CTRL/S and terminates the read using CTRL/Q.

/NOREADSYNC is the default; the system does not use CTRL/S and CTRL/Q to control reads to the terminal. /READSYNC is useful for certain classes of terminals that demand synchronization or on special-purpose terminal lines where data synchronization is appropriate.

/REMOTE
/NOREMOTE

Controls, for terminals attached to dial-up lines, whether the terminal can be accessed from a remote location.

The physical I/O privilege (PHY_IO) is required to use this qualifier.

If a terminal is in remote mode, it is reset to /NOREMOTE when the terminal is disconnected (that is, when you log out).

/SCOPE
/NOSCOPE

Indicates whether the terminal is a video terminal.

This qualifier is complementary to the /HARDCOPY qualifier, that is, /SCOPE is equivalent to /NOHARDCOPY.

/SPEED=rate

Specifies the rate at which the terminal sends and receives data.

You can specify rate as a single value to set the input and output baud rates to the same speed. To specify different baud rates for input and output, specify the rate in the format (n,m). The values n and m indicate the input and output baud rates, respectively.

The valid values for input and output baud rates are:

50	300	2400
75	600	3600
110	1200	4800
134	1800	7200
150	2000	9600

The default transmission rates are installation dependent.

/TAB
/NOTAB

Controls how the terminal handles tab characters. By default, the system expands all tab characters to blanks, assuming tab stops at 8-character intervals.

Use /TAB when you do not want the system to convert tabs to blanks, but want the terminal to process the tab characters.

/TTSYNC
/NOTTSYNC

Controls whether the terminal synchronizes output by responding to CTRL/S and CTRL/Q.

/TTSYNC is the default; the system stops sending output when you press CTRL/S and resumes output when you press CTRL/Q.

/TYPE_AHEAD
/NOTYPE_AHEAD

Controls whether the terminal accepts unsolicited input (that is, input that you type when there is no outstanding read).

When you specify /NOTYPE_AHEAD, the terminal is dedicated, and will only accept input when a program or the system issues a read to it.

Use this qualifier to ensure that a specific terminal remains dedicated to a particular application.

/UNKNOWN
Indicates that the terminal is of an unknown terminal type. When you specify this qualifier, default terminal characteristics for terminals of an unknown type are set. For a summary of the settings, see Table 3.

/UPPERCASE
/NOUPPERCASE
Specifies whether or not the terminal should translate all lowercase letters to uppercase.

This qualifier is complementary to the /LOWERCASE qualifier, that is, /UPPERCASE is equivalent to /NOLOWERCASE.

/VT05
Indicates that the terminal is a VT05 terminal. When you specify this qualifier, default terminal characteristics for VT05 terminals are set. For a summary of the settings, see Table 3.

/VT52
Indicates that the terminal is a VT52 terminal. When you specify this qualifier, default terminal characteristics for VT52 terminals are set. These settings are summarized under "VT5x" in Table 3.

/VT55
Indicates that the terminal is a VT55 terminal. When you specify this qualifier, default terminal characteristics for VT55 terminals are set. These settings are summarized under "VT5x" in Table 3.

/WIDTH=n
Specifies the number of characters on each input or output line. The width, n, must be in the range of 0 through 255.

If /WRAP is in effect, the terminal generates a carriage return/line feed when a line reaches the specified width.

/WRAP
/NOWRAP
Controls whether or not the terminal generates a carriage return/line feed when it reaches the end of the line. The end of a line is determined by the setting of the terminal width.

If you specify /NOWRAP, the terminal does not generate a carriage return/line feed when it reaches the end of a line, but continues to accept input at the last physical character position on the terminal line.

Examples

1. `$ SET TERMINAL/VT52`

   This SET command establishes the current terminal as a VT52 terminal and sets the default characteristics for that terminal type.

2.
```
$ SET    TERMINAL/WIDTH=132/PAGE=66/NOBROADCAST
$ TYPE    MEMO.DOC
     .
     .

$ SET    TERMINAL/LA36
```

This SET TERMINAL command indicates that the width of terminal lines is 132 characters and that the size of each page is 66 lines. The /NOBROADCAST qualifier disables the reception of broadcast messages while the terminal is printing the file MEMO.DOC. The next SET TERMINAL command restores the terminal to its default state.

Table 3
Default Characteristics for Terminals

Name Qualifier	UNKNOWN /UNKNOWN	LA36 /LA36	VT05 /VT05	VT5x /VT52 /VT55
BROADCAST	*	*	*	*
CRFILL	*	0	0	0
ECHO	*	yes	yes	yes
EIGHT_BIT	*	no	no	no
ESCAPE	*	*	no	*
HOLD_SCREEN	*	no	no	*
HOSTSYNC	*	no	no	yes
LFFILL	*	0	0	0
LOWERCASE	*	yes	no	yes
PAGE	*	*	*	*
PARITY	*	no	no	no
PASSALL	*	no	no	no
READSYNC	*	no	no	no
REMOTE	*	*	*	*
SPEED	*	*	*	*
TAB	*	no	no	yes
TTSYNC	*	yes	yes	yes
TYPE_AHEAD	*	yes	yes	yes
WIDTH	*	132	72	80
WRAP	*	yes	yes	yes

* Indicates that the current setting is not changed by the qualifier.

SET VERIFY

Controls whether command lines in command procedures are displayed at the terminal or printed in a batch job log.

Format

```
SET [NO]VERIFY

   Command Qualifiers

   None.
```

Prompts

None.

Command Parameters

None.

Description

By default, when the system processes command procedures executed interactively, it does not display the command lines at the terminal. System responses and error messages are always displayed.

If you use the SET VERIFY command to override the default setting, the system displays all the lines in command procedures as it executes them. If any lines contain lexical functions or symbol names that are substituted before command execution, the command interpreter displays the line as it appears after symbol substitution.

When you change the verification setting, it remains in effect for all command procedures that you subsequently execute.

The default setting for a batch job is VERIFY; that is, all lines in the command procedure appear in the batch job listing.

Examples

1.
```
$ SET VERIFY
    .
    .
$ SET NOVERIFY
$ EXIT
```

The verification setting is turned on for the execution of a command procedure. The system displays all the lines in the procedure, including command lines, as it reads them. At the end of the procedure, the SET NOVERIFY command restores the system default.

2.
```
$ VERIFY = 'F$VERIFY()
$ SET NOVERIFY
   .
   .
$ IF VERIFY THEN SET VERIFY
```

This command procedure uses the lexical function F$VERIFY to save the current setting of verification in the symbol named VERIFY (the function returns a value of 1 if verification is set on, a value of 0 if verification is set off). Then, the SET NOVERIFY command turns off verification. Subsequently, the IF command tests the value of VERIFY; if true (1), then verification is restored; if false (0), then verification remains off.

SET WORKING_SET

Redefines the default working set size for the process or sets an upper limit to which the working set size can be changed by an image that the process executes.

Format

```
SET WORKING_SET

    Command Qualifiers

    /LIMIT[=n]
    /QUOTA[=n]
```

Prompts

None.

Description

A process's working set is the number of pages that are resident in physical memory when an image is executing in the process. Each user is assigned a default working set size to be associated with the process created during login. The maximum size to which any process can increase its working set is defined in the user authorization file.

Command Qualifiers

/LIMIT[=n]
Specifies the maximum number of pages that can be resident in the working set during image execution.

The value, n, must be greater than the minimum working set defined at system generation and it must be less than or equal to the authorized limit defined in the user authorization file.

If you specify a value greater than the authorized limit, the command sets the working set limit at the maximum authorized value.

If you specify a value greater than the current quota, the quota value is also increased.

/QUOTA[=n]
Specifies the maximum number of pages that any image executing in the process can request. An image can set the working set size for the process by calling the Adjust Working Set Limit system service.

If you specify a quota value that is greater than the authorized quota, the working set quota is set to the authorized quota value.

Examples

1.
```
$ SHOW WORKING_SET
  Working Set    /Limit=100  /Quota=200  /Authorized Quota=200
$ SET WORKING_SET  /QUOTA=100
  New Working Set /Limit=100  /Quota=100
```

   The SHOW WORKING_SET command displays the current limit, quota, and authorized quota. The SET WORKING_SET command sets a quota limiting the maximum number of pages any image can request.

2.
```
$ SET WORKING_SET /LIMIT=200
  New Working Set /Limit=200  /Quota=200
```

   The SET_WORKING SET command sets both the working set size and the quota allowed to any image in the process to 200.

SHOW

Displays information about the current status of the process, the system, or devices in the system.

Format

```
SHOW    option
```

Options

```
[DAY]TIME
DEFAULT
DEVICES
LOGICAL
MAGTAPE
NETWORK
PRINTER
PROCESS
PROTECTION
QUEUE
RMS_DEFAULT
STATUS
SYMBOL
SYSTEM
TERMINAL
TRANSLATION
WORKING_SET
```

Prompts

What: option

Description

The SHOW command options are summarized in Table 4. Each SHOW command option and the format of the information it displays is described separately following Table 4.

Table 4
SHOW Command Options

Option	Displays
DAYTIME TIME	The current date and time
DEFAULT	The current default device and directory
DEVICES	The status of devices in the system
LOGICAL	Current logical name assignments
MAGTAPE	The status and characteristics of a specific magnetic tape device
NETWORK	The availability of network nodes, including the current node
PRINTER	Default characteristics of a line printer
PROCESS	Attributes of the current process, including privileges, resource quotas, memory usage, priority, and accounting information
PROTECTION	The current default protection applied to files
QUEUE	Printer or batch jobs that have been queued but have not completed
RMS_DEFAULT	The current default multi-block and multi-buffer counts used by RMS for file operations
STATUS	The status of the current job, including accumulated CPU time, open file count, and count of I/O operations
SYMBOL	Current symbol definitions
SYSTEM	A list of all processes in the system
TERMINAL	The device characteristics of a terminal
TRANSLATION	The result of translating a logical name
WORKING_SET	The current working set size limit and quota

SHOW DAYTIME

Displays the current date and time in the default output stream.

Format

```
SHOW [DAY]TIME

   Command Qualifiers

   None.
```

Prompts

None.

Example

1.
```
$ SHOW DAYTIME
  18-JAN-1977 00:03:45
```

   The SHOW DAYTIME command requests a display of the current date and time.

SHOW DEFAULT

Displays the current default device and directory name. These defaults are applied whenever you omit a device and/or directory name from a file specification.

The default disk and directory are established in the authorization file. You can change them during a terminal session or in a batch job with the SET DEFAULT command, or by reassigning the logical name SYS$DISK.

Format

```
SHOW DEFAULT

   Command Qualifiers

   None.
```

Prompts

None.

Examples

1.
```
$ SHOW DEFAULT
  DBA1:[ALPHA]
$ SET DEFAULT DBA2:[HIGGINS.SOURCES]
$ SHOW DEFAULT
  DBA2:[HIGGINS.SOURCES]
```

The SHOW DEFAULT command requests a display of the current default device and directory. The SET DEFAULT command changes these defaults, and the next SHOW DEFAULT command displays that the defaults have in fact been changed.

2.
```
$ ASSIGN DBA3:  SYS$DISK
$ SHOW DEFAULT
  DBA3:[HIGGINS2]
```

The ASSIGN command changes the equivalence name for the logical name SYS$DISK. This also changes the device name default, as the response from the SHOW DEFAULT command indicates.

SHOW DEVICES

Displays the status of all devices in the system, the status of a particular device, or lists the devices that currently have volumes mounted on them and/or are allocated to processes.

Format

```
SHOW DEVICES    [device-name]

    Command Qualifiers

    /ALLOCATED
    /BRIEF                      /BRIEF
    /FULL                       /BRIEF
    /MOUNTED
```

Prompts

None.

Parameters

device-name

Specifies the name of a device for which information is to be displayed. You can specify a complete device name or only a portion of a device name; the SHOW DEVICES command provides defaults for non-specified portions of device names, as follows:

- If you truncate a device name, for example if you specify "D", the command lists information about all devices whose device names begin with D.
- If you omit a controller designation, the SHOW DEVICES command lists all devices on all controllers with the specified unit number.
- If you omit a unit number, the SHOW DEVICES command lists all devices on the specified controller.

If you specify the SHOW DEVICES command and specify neither a device name parameter nor any qualifier, the command provides a brief listing of characteristics of all devices in the system. To obtain information about a specific device or generic class of devices, specify a device name.

Use the /ALLOCATED or /MOUNTED qualifier for a list of devices that are currently allocated to processes or mounted, respectively.

Command Qualifiers

/ALLOCATED

Requests a display of all devices currently allocated to processes.

If you specify a device name, the characteristics of only that device are displayed; if the device is not currently allocated the command displays a message indicating that there is no such device. If you specify a generic device name, the characteristics of all allocated devices of that type are displayed.

/BRIEF

Requests a brief display of information about the device(s) specified.

/FULL

Requests a complete listing of information about the device(s).

/MOUNTED

Requests a display of all devices that currently have volumes mounted on them.

If you specify a device name, only the characteristics of that device are displayed; however, if the device is not currently mounted, the command issues a message indicating there is no such device. If you specify a generic device name, the characteristics of all devices of that type that currently have volumes mounted are displayed.

Examples

1.
```
$ SHOW DEVICES
 List of Devices          on    19-JUN-1978 09:45:42.86
 Device Device      Device       Err.    Volume          Free  Trans Mount
 Name   Status  Characteristics Count     Label         Blocks Count Count
 DMA0:  on line  MNT ALL             1  AARDVARK          2938     1     1
 DMA2:  on line  MNT ALL             0  BACKUPC          20196     2     1
 OPA0:  on line                      0
 DXA1:  on line  MNT FOR             0  CONSOLE              0     1     1
 CRA0:  on line                      0
 LPA0:  on line  SPL ALL             0
 LPB0:  on line  ALL                 0
 TTA0:  on line                      0
 TTA1:  on line                      0
 TTA2:  on line                      0
   .
   .
 TTH7:  off line                     0
 XMA0:  on line                      0
 DBA1:  on line  MNT                 6  SYSTEMUSERS1   17397    14     1
 DBA2:  on line  MNT                 0  SYSTEMUSERS2   52630    18     1
```

This command displays, for each device in the system:

- Device name
- Device status (indicates whether the device is on line)
- Device characteristics (indicates whether the device is allocated or spooled, has a volume mounted on it or has a foreign volume mounted on it)
- Error count
- Volume label (for disk and tape volumes only)
- Number of free blocks on the volume
- Transaction count
- Number of mount requests issued for the volume (disk devices only)

2. $ SHOW DEVICES DMA0:/FULL

```
Device   DMA0:          19-JUN-1978 09:45:44.00
         on line
         Mounted
         Error Logging Enabled
         Allocated
 Error count:                    1    Owner process id          00010020
 Operations completed:          38    Owner process name        FACTOR
 Reference count:                1    Default buffer size            512

 Volume label       AARDVARK          Free blocks                  2938
 Owner UIC          [001,001]          Transaction count              1
 Volume protection  FF00               Mount count                    1
 Volume status                         Relative volume no.            0
 ACP process name   DBB2ACP            Cluster size                   2
                                       Max. files allowed          4000
```

The SHOW DEVICES command requests a full listing of the status of the RK06/RK07 device DMA0; the information indicates:

- Date and time of day
- Device status
- Error count
- Number of I/O operations completed
- Reference count
- Process identification of the owner of the device
- Process name of the owner of the device
- Default buffer size

For devices with volumes mounted on them, the command displays:

- Volume label
- User identification of the owner of the volume
- Protection code assigned to the volume
- Volume status (indicates whether it is mounted /SYSTEM, /GROUP, /SHARE, or /NOSHARE)
- Name of the Ancillary Control Process (ACP)
- Relative volume number
- Default cluster size
- Maximum number of files allowed on the volume

For tape devices, the command also displays the default density, the real volume label, and the recordsize.

SHOW LOGICAL

Displays all logical names in one or more logical name tables; or displays the current equivalence name assigned to a specified logical name by the ASSIGN, ALLOCATE, DEFINE, or MOUNT commands.

Format

```
SHOW LOGICAL    [logical-name]

   Command Qualifiers      Default

   /ALL                    /ALL
   /GROUP
   /PROCESS
   /SYSTEM
```

Prompts

None.

Command Parameters

logical-name

Specifies a 1- to 63-alphanumeric character logical name for which the equivalence name is to be displayed. The logical name is translated recursively a maximum of 10 times. For each translation, the process, group, and system logical name tables are searched, in that order, and the equivalence name for the first match found is displayed.

If you do not specify a logical name, the command displays all logical names in one or more tables, based on the presence of the /PROCESS, /GROUP, or /SYSTEM qualifiers. If no qualifiers are present and no logical is specified, the command displays all logical names in all logical name tables.

Command Qualifiers

/ALL

Specifies that all logical names in the specified logical name table(s) be displayed. If none of the qualifiers /PROCESS, /GROUP, or /SYSTEM is specified, all names in all logical name tables are displayed.

/GROUP

Indicates, when a logical-name parameter is present, that only the group logical name table is to be searched.

If you specify /ALL either explicitly or by default, all entries in the group logical name table are displayed.

/PROCESS

Indicates, when a logical-name parameter is specified, that only the process logical name table is to be searched.

If you specify /ALL either explicitly or by default, all entries in the process logical name table are displayed.

/SYSTEM
Indicates, when a logical-name parameter is present, that only the system logical name table is to be searched.

If you specify /ALL either explicitly or by default, all names in the system logical name table are displayed.

Examples

1. $ SHOW LOGICAL/PROCESS

```
Contents of process logical name table:

  SYS$INPUT = _TTB1:
  SYS$OUTPUT = _TTB1:
  SYS$ERROR = _TTB1:
  SYS$DISK = _DBA3:
  SYS$COMMAND = _TTAB1:
```

The SHOW LOGICAL command requests a display of the current process logical names. These are the default logical name assignments made by the command interpreter for an interactive process.

2. $ SHOW LOGICAL INFILE

```
  INFILE = DMB1:PAYROLL.DAT (group)
```

The SHOW LOGICAL command requests a display of the current equivalence name for the logical name INFILE. The response indicates that the logical name was found in the group logical name table.

3. $ SHOW LOGICAL/GROUP

```
Contents of group logical name table:

Group logical name table is empty
```

The SHOW command requests a display of all current logical names in the group logical name table. The message displayed indicates that there are no logical names in the group logical name table.

4. $ SHOW LOGICAL/SYSTEM SYS$LIBRARY

```
  SYS$LIBRARY   =   DBB2:[SYSLIB]   (system)
```

The SHOW LOGICAL command requests the equivalence name of SYS$LIBRARY. The response indicates that the default system libraries are in DBB2:[SYSLIB].

5. $ SHOW LOGICAL/GROUP/SYSTEM SYS$DISK

```
  SYS$DISK   =   DBA3:[1,1]   (system)
```

The SHOW LOGICAL command is qualified by both the /GROUP and /SYSTEM qualifiers; the response indicates that the logical name SYS$DISK has an equivalence name in the system logical name table.

SHOW MAGTAPE

Displays the current characteristics and status of a specified magnetic tape device.

Format

```
SHOW MAGTAPE    device-name

    Command Qualifiers

    None.
```

Prompts

Device: device-name

Command Parameters

device-name
Specifies the name of the magnetic tape device for which you want to display the characteristics and status.

Example

1.
```
$ SHOW MAGTAPE MTA0:

  MTA0: UNKNOWN, DENSITY=800, FORMAT=Normal-11 Odd Parity
```

The SHOW MAGTAPE command requests a display of the characteristics of the device MTA0. It displays the device type, density, and format (default or normal PDP-11).

It can also display the following characteristics:

Position Lost	Write-Locked
End-of-Tape	Even Parity
End-of-File	Odd Parity
Beginning-of-Tape	

SHOW NETWORK

Displays the availability of the local node as a member of the network and the names of all nodes that are currently accessible by the local node.

Format

```
SHOW NETWORK

   Command Qualifiers

   None.
```

Prompts

None.

Command Parameters

None.

Example

1.
```
$ SHOW NETWORK

                NETWORK STATUS AS OF 06-JUN-1978 12:42

                   LOCAL NODE NAME: VAX1
                   NUMBER:  3
                   STATE:  ON

                   NODE     LINE
                   MANILA XMAO
                   CHI      XMCO
```

The SHOW NETWORK command displays the name, number, and status of the local node and lists available remote nodes.

If no remote nodes are available, the command displays:

```
NO REMOTES ACCESSIBLE
```

If the network is unavailable, the command displays:

```
NETWORK UNAVAILABLE
```

SHOW PRINTER

Displays the default characteristics currently defined for a system printer.

Format

```
SHOW PRINTER    [device-name]

    Command Qualifiers

    None.
```

Prompts

Device: device-name

Command Parameters

device-name
Specifies the name of the printer for which characteristics are to be displayed.

Example

1.
```
$ SHOW PRINTER LPA0:

LPA0: LP11, WIDTH=132, PAGE=64, NOCR, FF, LOWERCASE
Device spooled to DBB2:
```

The SHOW PRINTER command requests a display of the characteristics of the printer LPA0.

SHOW PROCESS

Displays information about the current process.

Format

```
SHOW PROCESS

   Command Qualifiers

   /ACCOUNTING
   /ALL
   /PRIVILEGES
   /QUOTAS
   /SUBPROCESSES
```

Prompts

None.

Command Qualifiers

/ACCOUNTING
Displays accumulated accounting statistics for the current terminal session.

/ALL
Displays all information available, that is, the default information as well as the information displayed by the /ACCOUNTING, /PRIVILEGES, /QUOTAS, and /SUBPROCESSES qualifiers.

/PRIVILEGES
Displays the user privileges that have been granted to the process.

/QUOTAS
Displays the process's current quotas. The values displayed reflect any quota reductions resulting from subprocess creation.

/SUBPROCESSES
Displays the process name(s) of any subprocesses owned by the current process; if a hierarchy of subprocesses exists, the command displays the names in hierarchical order.

Examples

1. $ SHOW PROCESS

```
19-JUN-1978 11:59:44.13          _TTF3:        User : MALCOLM
Pid : 00160030    Proc. name : MALCOLM          UIC  : [122,001]
Priority :   4    Default file spec. :    DBA1:[MALCOLM.TESTFILES]

Devices allocated :   TTF3:
```

The default output of the SHOW PROCESS command displays:

- Date and time the SHOW PROCESS command is issued
- Device name of the current SYS$INPUT device
- User name
- Process identification number
- Process name
- User identification code (UIC)
- Base execution priority
- Default device
- Default directory
- Devices allocated to the process and volumes mounted, if any

2. $ SHOW PROCESS/ACCOUNTING

```
19-JUN-1978 11:59:44.54          _TTF3:       User : MALCOLM

Accounting information:

  Buffered I/O count :       2191        Peak working set size :       180
  Direct I/O count :          263        Peak virtual size :           196
  Page faults :              3131        Mounted volumes :               0
  Elapsed CPU time :      0 00:00:25.67
  Connect time :          0 01:17:47.76
```

3. $ SHOW PROCESS/PRIVILEGES

```
19-JUN-1978 11:59:44.71          _TTF3:       User : MALCOLM

Process privileges :

  GRPNAM       may insert in group logical name table
  GROUP        may affect other processes in same group
  PRMCEB       may create permanent common event clusters
  PRMMBX       may create permanent mailbox
  TMPMBX       may create temporary mailbox
```

4. $ SHOW PROCESS/QUOTAS

```
19-JUN-1978 11:59:44.86          _TTF3:       User : MALCOLM

Process Quota:

  Account name: DOCUMENT
  CPU limit :                 0 00:00:00.00    Direct I/ O limit :        6
  Buffered I/O byte count quota :    12480     Buffered I/O limit:        6
  Timer queue entry quota :             10     Open file quota :         16
  Paging file quota :              2560000     Subprocess quota :         8
  Default page fault cluster :         127     AST limit :               16
```

5. $ SHOW PROCESS/SUBPROCESSES

```
19-JUN-1978 11:59:45.41          _TTF3:        User : MALCOLM

Subprocesses owned :

- ORION
- CYGNUS
  - LYRA
```

SHOW PROTECTION

Displays the current file protection to be applied to all new files created during the terminal session or batch job. You can change the default protection at any time with the SET PROTECTION command.

Format

```
SHOW PROTECTION

   Command Qualifiers

   None.
```

Prompts

None.

Example

1.
```
$ SHOW PROTECTION
  SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
$ SET PROTECTION=(GROUP:RWED,WORLD:RE)/DEFAULT
$ SHOW PROTECTION
  SYSTEM=RWED, OWNER=RWED, GROUP=RWED, WORLD=RE
```

The SHOW PROTECTION command requests a display of the current protection defaults; the SET PROTECTION command changes the file access allowed to other users in the same group and to miscellaneous system users. The next SHOW PROTECTION command shows the modified protection defaults.

SHOW QUEUE

Displays the current status of entries in the printer and/or batch job queues.

Format

```
SHOW QUEUE    [queue-name]

    Command Qualifiers

    /ALL
    /BATCH
    /BRIEF
    /DEVICE
    /FULL
```

Prompts

Queue: queue-name

Command Parameters

queue-name
Specifies the name of a queue you want to display. The queue-name parameter is required if you do not specify either /BATCH or /DEVICE.

Command Qualifiers

/ALL
Displays the names of all jobs in the specified queue. By default, the SHOW QUEUE command displays only current jobs and pending jobs owned by the current process.

/BATCH
Displays entries in all batch job queues.

/BRIEF
Requests a brief listing of information about jobs in the queue. When you specify /BRIEF, only the user name, job number and queue name are displayed.

/DEVICE
Displays the status of jobs in all device queues.

/FULL
Displays the file specifications of each file in each pending job in the queue.

Examples

1.
```
$ SHOW QUEUE /DEVICES

* Device Queue SYS$PRINT Forms=0, Gendev Flag

  Holding Job  261 MALCOLM      BETA    , Pri=4, 19-JUN-1978 12:56

* Device Queue LPA0 Forms=0, Genprt Lower Flag

  Current Job  260 CRAMER       ALPHA   , Pri=4, 19-JUN-1978 12:55
  Pending Job  261 HIGGINS      TEMP0   , Pri=4, 19-JUN-1978 12:59
  Pending Job  262 HIGGINS      TEMPB   , Pri=4, 19-JUN-1978 13:05

* Device Queue LPB0 Forms=0, Genprt Flag
```

The SHOW QUEUE command displays the status of the printer queues. The first queue, named SYS$PRINT, consists of jobs that are being held. The printer queue, LPA0, is currently processing a job for the user CRAMER; two jobs are pending for the user HIGGINS (who issued the command). There are no jobs in the queue LPB0.

2.
```
$ SHOW QUEUE SYS$BATCH/FULL

* Batch queue SYS$BATCH Joblim=6, Inipri=4, Swap

  Current Job  263 MALCOLM      SLEEP    Pri=4, 19-JUN-1978 13:08
  Current Job  261 HIGGINSB     WAITF    Pri=4, 19-JUN-1978 13:02
  Current Job  260 CASEY        RECORD   Pri=4, 19-JUN-1978 12:59
  Current Job  259 MALCOLM      BATCH1   Pri=4, 19-JUN-1978 12:58
  Current Job  258 CLAUDIUS     XRULE    Pri=4, 19-JUN-1978 12:58
  Current Job  257 HAPPY        CALC     Pri=4, 19-JUN-1978 12:57
  Holding Job  262 HIGGINS      PROCEDUR Pri=4, 19-JUN-1978 13:11
    DBA1:PROCEDURE.COM;26
  Pending Job 265 HIGGINS       BATCHAVE Pri=4, 19-JUN-1978 13:12
      [1 Intervening Jobs]
  Pending Job  267 HIGGINS      ATTN     Pri=4, 19-JUN-1978 16:59
    DB1:T.COM;1  Delete
```

The SHOW QUEUE command requests a display of all jobs in the batch job queue. The /FULL qualifier requests the file specifications of pending files in the job. The response indicates a held job and two pending jobs for the user HIGGINS. The job ATTN, consisting of the file T.COM, is marked for deletion after processing.

SHOW RMS_DEFAULT

Displays the current default multi-block count and multi-buffer count that VAX-11 RMS uses for file operations.

Format

```
SHOW RMS_DEFAULT
```

Command Qualifiers

None.

Prompts

None.

Command Parameters

None.

Example

1.
```
$ SHOW RMS_DEFAULT
                MULTI-    |              MULTI-BUFFER COUNTS
                BLOCK     | Indexed  Relative          Sequential
                COUNT     |                     Disk   Magtape  Unit Record
Process          16       |    0         0        0        8        0
System            4       |    0         0        0        0        0
```

The SHOW RMS_DEFAULT command displays the current process and system default multi-block and multi-buffer counts for all types of file.

SHOW STATUS

Displays the status of the image currently executing in the process, if any. The SHOW STATUS command does not affect the image; you can continue the execution of the image after displaying its status.

Format

```
SHOW STATUS

   Command Qualifiers

   None.
```

Prompts

None.

Example

1.
```
$ RUN MYPROG
      .
      .
      .
^C
$ SHOW STATUS
  Status on  19-JUN-1978 13:16:32.20     Elapsed CPU :    0 00:00:18.37
  Buff. I/O :    1544    Cur. ws. :     180      Open files :          2
  Dir. I/O :      143    Phys. Mem. :    67      Page Faults :      2851
```

The RUN command executes the image MYPROG.EXE. While the program is running, CTRL/C interrupts it, and the SHOW STATUS command displays its current status.

- Current time and date
- Elapsed CPU time used by the current process
- Number of page faults
- Open file count
- Buffered I/O count
- Direct I/O count
- Current working set size
- Current amount of physical memory occupied

SHOW SYMBOL

Displays the current value of a local or global symbol. Symbols are defined with assignment statements (= command), by passing parameters to a command procedure file, or by the INQUIRE or READ commands.

Format

```
SHOW SYMBOL    [symbol-name]

    Command Qualifiers

    /ALL
    /GLOBAL
    /LOCAL
```

Prompts

Symbol: symbol-name

Command Parameters

symbol-name
Specifies the 1- to 255-alphanumeric character name of the symbol whose value you want to display. The symbol-name must begin with an alphabetic character. The SHOW SYMBOL command searches the local symbol table for the current command level, then local symbol tables for preceding command levels, then the global symbol table, for the specified symbol and displays the first match it finds.

If you specify /ALL, you cannot specify a symbol-name.

Command Qualifiers

/ALL
Requests that the current values of all symbols in the specified symbol table be displayed. If you specify /ALL and do not specify either /LOCAL or /GLOBAL, THE SHOW SYMBOL command displays the contents of the local symbol table for the current command level.

/GLOBAL
Requests that only the global symbol table be searched for the specified symbol name.

If you specify /ALL, all names in the global symbol table are displayed.

/LOCAL
Requests that only the local symbol table for the current command level be searched for the specified symbol name.

If you specify /ALL, all names in the local symbol table for the current command level are displayed.

Examples

1.
```
$ SHOW SYMBOL PRINT
  PRINT = PRINT/HOLD
```

   The SHOW SYMBOL command requests that the current value of the symbol name PRINT be displayed. The command interpreter searches the local symbol table for the current command level then local symbol tables for preceding command levels, then the global symbol table.

2.
```
$ SHOW SYMBOL/GLOBAL/ALL
  TIM = SHOW TIME
  LOG = @LOG
  $STATUS = %X00000001
  $SEVERITY = 1
```

   The SHOW SYMBOL command requests a display of all symbols defined in the global symbol table. Note that the symbols $STATUS and $SEVERITY, which are maintained by the system, are also displayed.

3.
```
$ SHOW SYMBOL/LOCAL TIM
  TIM =
```

   The SHOW SYMBOL command requests that only the local symbol table be searched for the symbol named TIM. The response indicates that TIM currently has no value.

SHOW SYSTEM

Displays a list of processes in the system and information about the status of each.

Format

```
SHOW SYSTEM

    Command Qualifiers

    None.
```

Prompts

None.

Example

1.
```
$ SHOW SYSTEM
   VAX/VMS  Processes on          19-JUN-1978 15:53:00.71
    Pid     Process Name     UIC  State Pri Dir. I/O    CPU      Page flts Ph.Mem
 00010000 NULL             000,000 COM    0        0 02:54:03.07         0    0
 00010001 SWAPPER          000,000 HIB   16        0 00:03:25.47         0    0
 00030017 CRAMER           150,020 LEF    4      265 00:00:11.32      1275   58
 00050019 ORION            124,001 COM    4       51 00:00:06.86       721  144 S
 0006001B DEBUG            274,010 LEFO   4       -- swapped out --      57
 00050023 _JOB350          262,020 COM    4     6272 00:01:08.82      6255  150 B
 0001003C OPERATOR         001,004 LEF   10       62 00:00:00.98        26   35
```

The response displays:

- Process identification
- Process name
- User identification code
- Process state
- Current priority
- Direct I/O count [1]
- Elapsed CPU time [1]
- Number of page faults [1]
- Physical memory occupied [1]
- Process indicator [2]

---

1 This information is displayed only if the process is currently in the balance set; if the process is not in the balance set, these columns contain the message:

```
-- swapped out --
```

2 The letter B indicates a batch job; the letter S indicates a subprocess; the letter N indicates a network process

SHOW TERMINAL

Displays the current characteristics of a specific terminal. Each of these characteristics can be changed with a corresponding option of the SET TERMINAL command.

Format

```
SHOW TERMINAL    [device-name]

   Command Qualifiers

   None.
```

Prompts

None.

Command Parameters

device-name
: Specifies the name of a terminal for which you want the characteristics displayed. If you do not specify a device name, the characteristics of the current device assigned to the logical name SYS$COMMAND are displayed.

Example

1.
```
$ SHOW TERMINAL

TTF3: /VT52, WIDTH=80, PAGE=24, OWNER=SELF
      SPEED=(2400,2400), CRFILL=0, LFFILL=0, NO PARITY
      INTERACTIVE, ECHO, TYPEAHEAD, NOESCAPE, NOHOSTSYNC, TTSYNC,
      LOWERCASE, TAB, WRAP, SCOPE, LOCAL, NOHOLDSCREEN,
      NOEIGHTBIT, BROADCAST, NOREADSYNC,
```

   The SHOW TERMINAL command displays the characteristics of the current terminal.

SHOW TRANSLATION

Searches the process, group, and system logical name tables, in that order, for a specified logical name and returns the equivalence name of the first match found.

Format

```
SHOW TRANSLATION    logical-name

   Command Qualifiers

   None.
```

Prompts

Log_Name: logical-name

Command Parameters

logical-name
Specifies a 1- to 63-alphanumeric character logical name for which you want to display the translation. The translation is not recursive.

Examples

1.
```
$ SHOW TRANSLATION PAYROLL
  PAYROLL = DMA1:[ACCOUNTS.WORKING]FACTOR1.DAT;37 (process)
```

The SHOW TRANSLATION command displays the current equivalence name of the logical name PAYROLL.

2.
```
$ ASSIGN DBA1:  DISK
$ ASSIGN/GROUP DBB3:  DISK
$ SHOW TRANSLATION DISK
  DISK = DBA1:  (process)
```

ASSIGN commands place entries for the logical name DISK in both the process and group logical name tables. The SHOW TRANSLATION command shows the logical name for the first entry it finds: the equivalence name placed in the process logical name table.

3.
```
$ RUN ORION
^Y
$ SHOW TRANSLATION TERMINAL
  TERMINAL = _TTF3:  (process)
$ CONTINUE
```

The RUN command executes the image ORION.EXE. After CTRL/Y interrupts the image, the SHOW TRANSLATION command displays a logical name assignment. The CONTINUE command resumes the execution of the image.

SHOW WORKING_SET

Displays the working set quota and limit assigned to the current process.

Format

```
SHOW WORKING_SET

    Command Qualifiers

    None.
```

Prompts

None.

Example

1.
```
$ SHOW WORKING_SET
Working Set    /Limit=100  /Quota=200  /Authorized  Quota=200
```

   The response to this command indicates that the current process has a working set limit of 100 pages, a quota of 200 pages, and that the current quota is equal to the authorized limit (200 pages).

SORT/RSX11

Invokes the PDP-11 SORT utility program to reorder the records in a file into a defined sequence and to create a new file of the reordered records.

For complete details on the qualifiers discussed below and additional information on how to define and control sort operations, see the PDP-11 SORT Reference Manual.

Format

```
SORT/RSX11   input-file-spec   output-file-spec

    Command Qualifiers          Default

    /DEVICE=device-name
    /KEY=(field,...)            /KEY=(CN1.length)
    /PROCESS=type               /PROCESS=RECORD
    /RSX11
    /SPECIFICATION=file-spec
    /WORK_FILES=n               /WORK_FILES=5


    File Qualifiers

    /ALLOCATION=n
    /BLOCK_SIZE=n
    /BUCKET_SIZE=n
    /CONTIGUOUS
    /FORMAT=(format,size)
    /INDEXED=keys
    /RELATIVE
    /SEQUENTIAL
```

Prompts

File: input-file-spec

Output: output-file-spec

Command Parameters

input-file-spec
Specifies the name of the file whose records are to be sorted. This file must be qualified with either the /FORMAT qualifier or the /INDEXED qualifier to indicate the precise format of the file.

output-file-spec
Specifies the name of the file into which the sorted records are to be written.

You can optionally qualify the output-file-spec parameter with the /FORMAT qualifier to indicate the desired output format.

Command Qualifiers

/DEVICE=device-name

Specifies the name of the device to be used for work files during sort execution, overriding the device specified when SORT was installed.

/KEY=field,...

Defines the fields in each input record that are the basis for the sort. This qualifier is required.

You can specify up to 10 key fields. Each key field must be specified in the format:

```
[a][b]m.n
```

a defines the way that the data is handled and interpreted. The valid keywords and their meanings are:

B	Two's complement binary
C	Alphanumeric
D	If alphabetic, numeric with superimposed sign If FORTRAN numeric, convert to binary
F	2- or 4-word floating point
I	As D above, but with leading + or - sign
J	As D above, but with trailing + or - sign
K	As D above, but with sign overpunched
P	Packed decimal
Z	ASCII zone

If not specified, the default is C, that is, the sort is alphanumeric.

b specifies the general sorting order. You can specify:

N	Ascending order
O	Descending order

If not specified, the default is N, that is, records are sorted in ascending order.

m is a decimal number defining the beginning position of the key field relative to the beginning of each record, with 1 indicating the first position in the record.

This field is required.

n specifies the size of the key field, in bytes.

This field is required.

If you specify more than one key field, separate the specifications with commas and enclose the list in parentheses.

/PROCESS=type
Defines the type of sort. You can specify one of the following options:

ADDRESS Requests that SORT produce an address file without reordering the input file.

INDEX Requests that SORT produce an index file containing the key field of each data record and a pointer to its location in the input file.

RECORD Requests SORT to sort the entire contents of each record in the input file.

TAG Requests SORT to sort only on the record keys of each record in the input file.

By default, the SORT/RSX11 command produces a record sort.

/RSX11
Requests the PDP-11 SORT program. This qualifier is required.

/SPECIFICATION=file-spec
Specifies the name of a file containing SORT-11 specifications to control the sorting process.

For details on the contents of this file, see the PDP-11 SORT Reference Manual.

/WORK_FILES=n
Defines the number of work files to be used during the sorting process, overriding the system-defined default.

You can specify from 3 to 8 work files. The default is 5.

File Qualifiers

/ALLOCATION=n
Specifies the number of 512-byte blocks to allocate for the output file. This qualifier can only be used to qualify the output file.

If no allocation quantity is specified, SORT-11 uses a default allocation quantity based on the type of sorting process.

/BLOCK_SIZE=n
Specifies, when the input and/or output file is a magnetic tape volume, the size of the blocks to be read or written. If not specified, the block size defaults to 512 bytes.

/BUCKET_SIZE=n
Specifies the RMS bucket size allocation for the output file. This qualifier can only be used to qualify the output file parameter.

If no bucket size is specified, SORT-11 uses the bucket size of the input file if the input and output file organizations are similar. If the input file organization is different than the organization requested for the output file, the default is 1.

/CONTIGUOUS

Requests the output file to be written into contiguously allocated disk space. This qualifier can only be used to qualify the output file parameter.

By default, SORT-11 does not create contiguous output files.

/FORMAT=(format,size)

Defines the format and record size of input and output files, where format is one of the keywords listed below and size is the length, in bytes, of the largest record in the file. The valid record formats are:

FIXED
VARIABLE
STREAM
UNKNOWN

This qualifier is required on the input file specification; if not specified for the output file, the output file format defaults to the format of the input file (for RECORD and TAG sort processing). For ADDRESS_ROUTING sort processing, the output file record size is 6 bytes; for INDEX sort processing, the output file record size is 6 bytes plus the size of the input record.

/INDEXED=keys

Specifies that the associated input file is an indexed sequential file and indicates the number of keys in each record in the file.

/RELATIVE

Requests the output file to be in relative file organization. By default, the output file has the same format as the input file.

/SEQUENTIAL

Requests the output file to be in sequential file organization. By default, the output file has the same format as the input file.

Examples

1.
```
$ SORT/RSX11  CUSTOMER.FIL/FORMAT=(FIXED,80) -
$_ALPHA.SRT/KEY=(1,20)
SRT -- M:ELAPSED REAL TIME: 00:00:17
SRT -- M:TOTAL RECORDS SORTED:      1684
```

The SORT command requests a default alphanumeric sort on the records in the file CUSTOMER.FIL. The SORT program sorts the records based on the contents of the first 20 characters in each record and writes the sorted list into the output file ALPHA.SRT.

2.
```
$ SORT/RSX11  CUSTOMER.FIL/FORMAT=(FIXED,80) -
$_TENURE.FIL/KEY=(29,2,26,2,23,2)
SRT -- M:ELAPSED REAL TIME: 00:00:25
SRT -- M:TOTAL RECORDS SORTED:      3245
```

The key fields specified for this SORT command request that the records be sorted first on the 2 characters beginning in column 29, then on the 2 characters beginning in column 26, then on the 2 characters beginning in column 23. If columns 23 through 30 of each record contain a date in the format:

dd-mm-yy

This command creates an output file with records sorted in ascending order of date.

STOP

Terminates execution of:

- A command, image, or command procedure that was interrupted by CTRL/Y
- A command procedure
- A subprocess or a detached process

Format

```
STOP    [process-name]

    Command Qualifiers

    /IDENTIFICATION=process-id
```

Prompts

None.

Command Parameters

process-name
Specifies the 1- to 15-alphanumeric character string name of the process to be deleted. The specified process must have the same group number in its user identification code (UIC) as the current process.

If you specify /IDENTIFICATION, the process name is ignored. If you specify neither the process-name parameter nor the /IDENTIFICATION qualifier, the image executing in the current process is terminated.

Description

The STOP command causes an abnormal termination of the image currently executing; if the image has declared any exit handling routines, they are not given control.

Note that when an image is interrupted by CTRL/Y, and the RUN command is issued to execute another image, the interrupted image is also terminated. However, in this case exit handling routines are allowed to execute before the next image is run.

If you interrupt a command procedure by CTRL/Y and you issue the STOP command, or if the STOP command is executed in a command procedure, all command levels are unstacked and control returns to the command interpreter.

If you specify a process name or process identification, the STOP command terminates the image currently executing in the specified process and deletes the process. If the process is a batch job process, no notification of deletion occurs; the log file for the batch job does not print.

The user privilege GROUP is required to stop other processes in the same group. The user privilege WORLD is required to stop any process in the system.

Command Qualifiers

/IDENTIFICATION=process-id

Specifies the process identification the system assigned to the process when the process was created. When you create a process with the RUN command, the RUN command displays the process identification number of the process it creates.

When you specify the process identification, you can omit leading zeros.

Examples

1.
```
$ RUN  MYPROG
^Y
$ STOP
```

The RUN command begins executing the image MYPROG. Subsequently, CTRL/Y interrupts the execution and the STOP command terminates the image.

2.
```
$ @TESTALL
^Y
$ STOP
```

The @ (Execute Procedure) command executes the procedure TESTALL.COM. CTRL/Y interrupts the procedure and the STOP command returns control to the DCL command interpreter.

3.
```
$ RUN/PROCESS_NAME=LIBRA  LIBRA
%RUN-S-PROC_ID, identification of created process is 0013340D
    .
    .
$ STOP LIBRA
```

The RUN command creates a subprocess named LIBRA to execute the image LIBRA.EXE. Subsequently, the STOP command forces the image to exit and deletes the process.

4.
```
$ ON ERROR THEN STOP
    .
    .
```

In a command procedure, the ON command establishes a default action when any error occurs as a result of a command or program execution. The STOP command stops all command levels; if this ON command is executed in a command procedure that is executed from within another procedure, control does not return to the outer procedure, but to the command interpreter.

SUBMIT

Enters a command procedure in the batch job queue.

Format

```
SUBMIT    file-spec,...

   Command Qualifiers              Default

   /AFTER=absolute-time
   /HOLD                           /NOHOLD
   /[NO]IDENTIFY                   /IDENTIFY
   /NAME=job-name
   /PARAMETERS=parameters,...
   /PRIORITY=n
   /QUEUE=queue-name
   /REMOTE

   File Qualifiers

   /DELETE                         /NODELETE
```

Prompts

File: file-spec,...

Command Parameters

file-spec,...

Specifies one or more command procedures to be submitted for batch job execution. You must specify a file name; if you do not specify a file type, the SUBMIT command uses the default file type of COM. If you specify more than one file, you can separate them either with commas (,) or plus signs (+).

If the file specification contains a network node name, the /REMOTE qualifier must be specified.

Description

A file or files queued by the SUBMIT command are considered a job. The system assigns a unique job identification (jobid) to each job in the system. When you submit a batch job, the system displays both the jobid it assigned to the job and the name of the batch job queue in which it entered your job.

Batch Job Output: When you submit command procedures for processing by the SUBMIT command, all output from the command procedure is written to a file called name.LOG where name is the file name of the first command procedure file in the job. (Use the /NAME qualifier to give the job a different name.) This file is initially written on your default disk; when the batch job completes, the system queues the file to SYS$PRINT and deletes the file after it has printed.

If multiple procedures are submitted, the job terminates if any procedure exits with an error or fatal error status.

Command Qualifiers

/AFTER=absolute-time
Requests that the job be held until after a specific time. If the specified time has already passed, the job is queued for immediate processing.

Specify the time value according to the rules for entering absolute times (these rules are given in Section 6.8).

/HOLD
/NOHOLD
Controls whether or not the job is to be made available for immediate processing.

If you specify /HOLD, the job is not released for processing until you specifically release it with the SET QUEUE/RELEASE command.

/IDENTIFY
/NOIDENTIFY
Controls whether the command interpreter displays the jobid assigned to the job and the name of the queue in which the job was entered.

By default, the jobid and queue name are displayed whenever a job is successfully queued.

/NAME=job-name
Defines a 1- to 8-alphanumeric character name string to identify the job. The job name is displayed by the SHOW QUEUE command, and is printed on the flag page of the batch job output log, replacing the file name of the log file.

If you do not specify /NAME, the name string defaults to the file name (truncated to 8 characters, if necessary) of the first, or only, file in the job.

/PARAMETERS=parameters
Specifies from 1 to 8 optional parameters to be passed to the job. The parameters define values to be equated to the symbols named P1, P2, P3, and so on, in each the command procedure in the job. The symbols are local to the specified command procedures.

If you specify more than one parameter, separate them with commas and enclose them in parentheses.

The commas delimit the parameters. To specify a parameter that contains any special characters or delimiters, enclose the parameter in quotation marks.

/PRIORITY=n
Specifies the priority for the specified job. The priority, n, must be in the range of 0 through 31, where 0 is the lowest priority and 31 is the highest.

By default, jobs are queued at the same priority as your current process priority; the user priviledge ALTPRI is required to set a priority value that is higher than your current process's priority.

/QUEUE=queue-name
Specifies the name of a specific batch job queue to which the job is to be submitted.

/REMOTE
Indicates that the specified command procedure be executed on a remote node. The file specification must contain the name of the node on which the file resides and at which the procedure is to be executed.

If you specify /REMOTE, you cannot specify any other qualifiers.

File Qualifiers

/DELETE
/NODELETE
Controls whether files are deleted after processing. If you specify /DELETE after the SUBMIT command name, all files in the job are deleted. If you specify /DELETE following a file specification, only the associated file is deleted after it is processed.

The protection code on the input file(s) must allow delete access to the default user identification code (UIC) of the user who submitted the job.

Examples

1.
```
$ SUBMIT AVERAGE
  Job 112 entered on queue SYS$BATCH
```

The SUBMIT command enters the procedure AVERAGE.COM in the batch job queue. When the batch job completes, the log file AVERAGE.LOG is queued for printing.

2.
```
$ SUBMIT  BACKUP/PARAMETERS=(TXT,DOC,MEM), -
$_AVERAGE, RUNMASTER
  Job 416 entered on queue SYS$BATCH
```

The SUBMIT command enters three command procedures in a single job. The job is given three parameters: P1 is equated to the string TXT, P2 to the string DOC and P3 to the string MEM. After the procedure BACKUP.COM is executed, the procedures AVERAGE.COM and RUNMASTER.COM are executed.

3.
```
$ SUBMIT/NAME=BATCH_24/HOLD TESTALL
  Job 467 entered on queue SYS$BATCH
```

The SUBMIT command enters the procedure TESTALL.COM for processing as a batch job, but in a HOLD status. The job will not be released until the SET QUEUE/RELEASE command is issued. The /NAME parameter requests that the batch job be identified as BATCH_24.

SYNCHRONIZE

Places the process executing a command procedure in a wait state until a specified batch job completes execution.

Format

```
SYNCHRONIZE    [job-name]

    Command Qualifiers

    /ENTRY=jobid
    /QUEUE=queue-name
```

Prompts

None.

Command Parameters

job-name
Specifies the 1- to 8-alphanumeric character string name of the batch job. The job-name corresponds to the name of the job defined by /NAME qualifier when the job was submitted or to the default job name assigned by the system.

The job must be associated with your current login user name.

If you specify /ENTRY, the job-name parameter is ignored. You must specify either a job name or a jobid.

Description

The SYNCHRONIZE command provides batch job synchronization. If a job specified is not currently in the system, the command completes immediately with a warning message.

Command Qualifiers

/ENTRY=jobid
Specifies the system-assigned job identification of the batch job. The system displays the job identification when it successfully queues a job for execution; the job identification of a batch job is also displayed when you issue the SHOW QUEUE command.

/QUEUE=queue-name
Specifies the name of the queue on which the job was entered. If not specified, the command assumes that the job is in the default batch job queue, SYS$BATCH.

The queue-name specified is subject to one level of logical name translation.

Example

1.
```
$ SUBMIT/NAME=PREP FORMAT/PARAMETERS=(SORT,PURGE)
$ SUBMIT PHASER
```

The first SUBMIT command submits the command procedure FORMAT.COM for execution and gives the job the job name PREP. The second SUBMIT command queues the procedure PHASER.COM. The procedure PHASER.COM contains the line:

```
$ SYNCHRONIZE PREP
```

When this line is processed, the system verifies whether the batch job named PREP is currently executing. If it is, the procedure PHASER is forced to wait until PREP completes execution.

TYPE

Displays the contents of a file or group of files on the current output device.

Format

TYPE file-spec,...	
Command Qualifiers	Default
/OUTPUT=file-spec	/OUTPUT=SYS$OUTPUT

Prompts

File: file-spec,...

Command Parameters

file-spec,...
Specifies one or more files to be displayed. If you specify a file name and do not specify a file type, the TYPE command uses the default file type of LIS.

If you specify more than one file, separate the file specifications with either commas (,) or plus signs (+). In either case, the files are displayed in the order listed.

You can specify wild cards in place of the file name, file type, or file version fields. The TYPE command displays all files that satisfy the file description.

Description

When the TYPE command displays output, you can control the display in any of the following ways:

- To temporarily halt the output, then resume it at the line at which it was interrupted, use CTRL/S followed by CTRL/Q.
- To suppress the display, but continue command processing, use CTRL/O. If you press CTRL/O again before the command terminates, output resumes at the current point in command processing. If you press CTRL/O when the TYPE command is displaying files in a list, the TYPE command suppresses typing the current file and begins typing the next file in the list.
- To stop command execution entirely, press CTRL/Y, then issue the STOP command or any other DCL command that terminates the image.

Command Qualifiers

/OUTPUT=file-spec
Requests that the output from the TYPE command be written to the specified file, rather than SYS$OUTPUT.

Examples:

1. ```
   $ TYPE  COMMON.DAT
   ```

   The TYPE command requests that the file COMMON.DAT be displayed at the terminal.

2. ```
   $ TYPE *.DAT
   .
   .
   .
   ^O
   .
   .
   .
   ^Y
   $ STOP
   ```

   The TYPE command contains a wild card in place of the file name. Files with file types of DAT are displayed; when CTRL/O is pressed, output of the current file stops and the TYPE command begins displaying the next file. When CTRL/Y interrupts the command, the STOP command terminates the TYPE command.

UNLOCK

Makes accessible a file that became inaccessible as a result of being improperly closed.

Format

```
UNLOCK    file-spec,...

   Command Qualifiers

   None.
```

Prompts

File: file-spec,...

Command Parameters

file-spec,...
Specifies one or more files to be unlocked. If you specify more than one file specification, separate them with either commas (,) or plus signs (+).

No wild cards are allowed in the file specifications.

Example

1.
```
$ TYPE TESTFILE.OUT
%TYPE-E-OPENIN, error opening DBA1:[MALCOLM]TESTFILE.OUT;3
as input
-SYSTEM-W-FILELOCKED, file is deaccess locked
$ UNLOCK TESTFILE.OUT
$ TYPE TESTFILE.OUT
```

The request to type the output file, TESTFILE.OUT, returns an error message that indicates the file is locked; the UNLOCK command unlocks it. The TYPE command is used to verify the contents of the file, which may be incomplete.

WAIT

Places the current process in a wait state until a specified period of time has elapsed. The WAIT command is provided for use in command procedures to delay processing of the procedure or of a set of commands in a procedure for a specific amount of time.

Format

```
WAIT    delta-time

    Command Qualifiers

    None.
```

Prompts

None.

Command Parameters

delta-time

Specifies the time interval to wait. The time must be specified according to the rules for specifying delta time values (these rules are given in Section 6.8). Note, however, that the delta time can contain only the hours, minutes, and seconds fields; the days part must be omitted.

Note that if you issue the WAIT command interactively, the WAIT command does not prompt; however, a time value is required.

Description

If you enter the WAIT command interactively, your current process is placed in a wait state and you cannot enter any more commands. (You can, however, receive unsolicited messages from other processes.) Press CTRL/C or CTRL/Y to restore normal terminal interaction.

Example

1.
```
$ LOOP:
$ RUN ALPHA
$ WAIT 00:10
$ GOTO LOOP
```

The command procedure executes the program image ALPHA. After the RUN command executes the program, the WAIT command delays execution of the next command for 10 minutes. After 10 minutes, the GOTO command executes and the procedure loops to the label LOOP and executes the program again. The procedure loops until interrupted or terminated.

If the procedure is executed interactively, it can be terminated by pressing CTRL/C or CTRL/Y and issuing the STOP command or another DCL command that runs a new image in the process. If the procedure is executed in a batch job, it can be terminated with the DELETE/ENTRY command.

WRITE

Writes a record to a specified output file.

Format

```
WRITE    logical-name  symbol-name,...

   Command Qualifiers

   /ERROR=label
```

Prompts

Log_Name: logical-name

Symbol: symbol-name,...

Command Parameters

logical-name
Specifies the logical name assigned to the file to which a record is to be written. The OPEN command assigns a logical name to a file and places the logical name in the process logical name table.

symbol-name,...
Specifies data to be written as a single record to the output file. You can specify one or more symbol names, character strings enclosed in quotation marks, or literal numeric values. The items specified in the symbol name list must be separated by commas; the command interpreter concatenates the items into a single record and writes the record to the output file.

The maximum size of any record that can be written is 255 bytes.

Description

The WRITE command can write records only to sequential files, and cannot be used to append new records at the end of existing files.

Before a file can be written, it must be opened with the OPEN command and assigned a logical name. The process permanent files identified by the logical names SYS$INPUT, SYS$OUTPUT, SYS$ERROR, and SYS$COMMAND do not have to be explicitly opened to be written.

Command Qualifiers

/ERROR=LABEL

Specifies a label on a line in the command procedure to receive control if the write request results in an error. If no error routine is specified and an error occurs during the writing of the file, the command procedure continues execution at the next line in the file, as it does if no error occurs.

The error routine specified for this qualifier takes precedence over any action statement indicated in an ON command. If /ERROR is not specified, the current ON condition action is taken.

If an error occurs and the target label is successfully given control, the reserved global symbol $STATUS contains a successful completion status value.

Examples

1.
```
$ WRITE SYS$OUTPUT "Beginning second phase of tests"
```

The WRITE command writes a single line of text to the current output device.

2.
```
$ OPEN/WRITE OUTPUT_FILE TESTFILE
$ INQUIRE ID "Assign Test-id Number"
$ WRITE/ERROR=END_LOOP  OUTPUT_FILE  "Test-id is ",ID
$ WRITE/ERROR=END_LOOP  OUTPUT_FILE  ""
$ !
$ WRITE_LOOP:
   .
   .

$ GOTO WRITE_LOOP
$ END_LOOP:
$ !
$ CLOSE OUTPUT_FILE
$ PRINT/DELETE TESTFILE.DAT
```

The OPEN command opens the file TESTFILE.DAT; the INQUIRE command requests an identification number to be assigned to a particular run of the procedure. The number entered is equated to the symbol ID. The WRITE commands write a text line concatenated with the symbol name ID and a blank line.

The lines between the label WRITE LOOP and END LOOP define processing that results in additional data written to the file. The label END_LOOP is also used as the target of the /ERROR qualifier on the WRITE command; the CLOSE and PRINT commands at this label close the output file and queue a copy of the file to the system printer. The output file TESTFILE.DAT is deleted after printing.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

VAX/VMS
Command Language
User's Guide
AA-D023A-TE

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 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

— — — Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

RT/C SOFTWARE PUBLICATIONS TW/A14
DIGITAL EQUIPMENT CORPORATION
1925 ANDOVER STREET
TEWKSBURY, MASSACHUSETTS 01876

— — Do Not Tear - Fold Here —