DataGeneral

Software Documentation

Command Line Interpreter (CLI) User's Manual (AOS and AOS/VS)

Command Line Interpreter (CLI) User's Manual (AOS and AOS/VS)

093-000122-07

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-000122

Revision 07, October 1984

NOTICE

Command Line Interpreter (CLI)
User's Manual
(AOS and AOS/VS)
093-000122

Revision History:

Original Release - April 1976
First Revision - April 1977
Second Revision - June 1978
Third Revision - June 1979
Fourth Revision - November 1980
Fifth Revision - May 1982
Sixth Revision - February 1984
Seventh Revision - October 1984

Effective with:

(AOS Rev. 6.00) (AOS/VS Rev. 5.00)

A vertical bar in the margin of a page indicates substantive change from the previous revision, except for Chapter 7, which is all new material.

Preface

This manual describes the Advanced Operating System (AOS) and Advanced Operating System/Virtual Storage (AOS/VS) Command Line Interpreter (CLI). It assumes that you have had some experience with AOS or AOS/VS whether on a DESKTOP GENERATION™ computer system or on the larger MV/Family or ECLIPSE® series of computers. The CLI utilities for AOS and AOS/VS are almost identical. In the few instances where a CLI command or command switch is valid for one operating system but not the other, we highlight the difference. If we do not specify otherwise, you can assume that we are speaking about both CLIs.

If you are new to AOS or AOS/VS, we recommend that you read *Learning to Use Your Advanced Operating System* (069-000018) or *Learning to Use Your AOS/VS System* (069-000031). These books lead you through a sample CLI session, list some common CLI commands, and supply essential background information for Data General's advanced operating systems.

Chapters 1 through 5 of this manual are a tutorial text. Chapter 6 "CLI Commands, Pseudo-Macros, Languages, and System Utilities" is a reference chapter.

Chapter 1 introduces the CLI, terminal keyboard and explains console control characters.

Chapter 2 outlines the AOS and AOS/VS file system and how to navigate through it with the CLI.

Chapter 3 describes CLI command line syntax.

Chapter 4 explains the CLI environment levels and how you can use them.

Chapter 5 explains CLI macros (user-defined instructions which execute a series of CLI commands).

Chapter 6 lists all CLI commands, pseudo-macros, languages and system utilities by category and alphabetically.

Chapter 7 summarizes how to use labeled magnetic tapes as well as labeled diskettes.

Appendix A lists the error messages.

Appendix B describes an interface that you can use in an assembly language program to communicate with the CLI.

Appendix C contains the ASCII character set.

Appendix D explains how to submit batch jobs to the stacker utility.

Appendix E summarizes the format necessary to access remote resources (networking) from the CLI. For more information about networking, see the *XODIAC™ Network Management System User's Manual* (093-000178).

At the end of the manual is a Glossary of technical terms used throughout the manual.

We also supply two tabbed divider pages with this manual. One sets off Chapter 6 to aid speedy access to this important reference chapter. The second one sets off Appendix A, the error messages appendix, also for speedy access.

Suggested Manuals

You can find many concepts introduced in this manual documented in greater depth in other AOS and AOS/VS manuals. In certain instances, you may need to refer to the following documents for further details:

Learning to Use Your Advanced Operating System (069-000018) and *Learning to Use Your AOS/VS System* (069-000031) provide a sample user's session on an AOS or AOS/VS timesharing system. Read the appropriate manual to see how to approach AOS or AOS/VS.

The AOS Programmer's Manual (093-000120) is an in-depth study of the Advanced Operating System. This manual is a primary reference for assembly language programmers who plan to use AOS. It explains AOS system calls, language processing concepts, memory management, file input/output, and multitasking.

Advanced Operating System/Virtual Storage (AOS/VS) Programmer's Manual is in two volumes. Volume 1 (093-000335) contains explanations of basic AOS/VS concepts and how families of system calls work together. Volume 2 (093-000241) contains the AOS/VS system calls. The system calls are arranged alphabetically for your convenience.

How to Generate and Run AOS (093-000217) describes how to generate and run AOS. Generating includes formatting one or more blank disks, installing a starter system, then creating the AOS multiuser environment. Running includes day-to-day operation: bringing up the AOS system, shutting it down, dumping files for backup, making decisions that help the system run more efficiently.

The *How to Generate and Run AOS/VS* (093-000243) describes how to generate and run AOS/VS. Generating includes formatting one or more blank disks, installing a starter system, bringing it up, generating a tailored system, then creating an AOS/VS multiuser environment. Running includes day-to-day operation: bringing up the AOS/VS system, shutting it down, dumping files for backup, and making decisions that help the system run most efficiently.

Reader, Please Note:

We use these conventions for command formats in this manual:

COMMAND required *[optional]* ...

Where	Means
COMMAND	You must enter the command (or its accepted abbreviation) as shown.
required	You must enter some argument (such as a filename). Sometimes, we use: $\{\text{required}_1 \mid \text{required}_2\}$ which means you must enter one of the arguments. Don't enter the braces; they only set off the choice.
[optional]	You have the option of entering this argument. Don't enter the brackets; they only set off what's optional.
...	You may repeat the preceding entry or entries. The explanation will tell you exactly what you may repeat.

Additionally, we use certain symbols in special ways:

Symbol	Means
⤶	Press the NEW LINE or carriage return (CR) key on your terminal's keyboard.
□	Be sure to put a space here. (We use this only when we must; normally, you can see where to put spaces.)

When we tell you to enter something, we mean that you are to press the NEW LINE or the carriage return (CR) key.

All numbers are decimal unless we indicate otherwise; e.g., 358.

And finally, in examples we use

THIS TYPEFACE TO SHOW YOUR ENTRY ⤶
THIS TYPEFACE FOR SYSTEM QUERIES AND RESPONSES.

) is the CLI prompt.

Contacting Data General

If you have comments on this manual, please use the prepaid Remarks Form that appears after the Index.

If you need additional manuals, please use the enclosed TIPS order form (USA only) or contact your local Data General sales representative.

End of Preface

Contents

Chapter 1 - Introduction

Chapter 2 - The File System

Chapter 3 - Command Line Syntax

Chapter 4 - CLI Environments and Exceptional Condition Handling

Chapter 5 - CLI Macros

Chapter 6 - CLI Commands, Pseudo-Macros, Languages and Systems Utilities

Chapter 7 - Using Magnetic Tape

Tables

Table

Illustrations

Figure

Chapter 1
Introduction

The Command Line Interpreter (CLI) is your primary communication link to the computer. It is an interactive programming language with built-in commands and pseudo-macros. Pseudo-macros are system commands that either return a value or create conditional expressions. (See the Glossary to check on any words you are unsure about.)

For most system configurations, the CLI will begin running as soon as you log on. The CLI commands give you access to most elements of your computer system. Using CLI commands, you can control peripheral devices, such as a line printer and a tape drive. (See Chapter 6.) You can also create, delete, and move files as well as execute programs and utilities. Because the CLI is an interpretive language, it will execute only one command at a time. However, using the CLI's special syntax (described in Chapter 3), you can enter a command once that will execute the same command a number of times with different arguments. Another handy feature is the CLI's macro capability (see Chapter 5). This allows you to place a number of CLI commands into a file and when you enter the name of the file, the CLI executes all of the commands.

Operating Your Terminal

In most instances, you will be issuing CLI commands from your terminal. Your terminal may be a DASHER® video display or a hard-copy terminal. In either case, it will have a keyboard with alphanumeric characters and function keys. Many of these function keys, however, are for other utilities and have no special use in the CLI. See the manual for each utility for more details on how it defines the function keys. Other function keys, which we describe in the next section of this chapter, perform special operations for the CLI.

Besides these special function keys, each keyboard has numeric, cursor control, and alphabetic keys.

Numeric Keypad

You can use both the numeric keypad at the far right of your keyboard or the numeric keys on the main keypad to enter digits.

Cursor Control Keypad

The cursor control keypad, not found on hard-copy terminals, is to the left of the numeric keypad. This keypad contains the keys that move the cursor around on the screen. Some of the keys on this keypad are the ERASE EOL, HOME, ERASE PAGE, and TAB keys, as well as all of the arrow keys.

ERASE EOL

The ERASE EOL key erases characters, beginning with the current cursor position, to the end of the line. (This key is not found on hard-copy terminals.)

HOME

On video display screens, HOME moves the cursor to the left margin of the current line.

ERASE PAGE

On video display terminals, the ERASE PAGE key will clear the screen of text and move the cursor to the top left-hand corner of the screen. This key also acts as a command terminator entering any characters that are on the line when you enter the ERASE PAGE key.

NOTE: If the page mode characteristic is on, ERASE PAGE may freeze the terminal. Type CTRL-Q to free it. You also use the ERASE PAGE key to change your password (see *Learning to Use Your Advanced Operating System* or *Learning to Use Your AOS/VS System* manual).

TAB

The TAB key moves the cursor to the next tab stop and inserts a tab character in the line. A tab occupies every 8 character positions so tab stops are in columns 9, 17, 25, 33, 41, 49, 57, 65, 73. Check each text editor's manual to see how each editor defines the TAB key.

The Arrow Keys

Within some system utilities and languages, the up arrow (↑) key moves the cursor up one line while remaining in the same column.

Within some system utilities and languages, the down arrow (↓) key moves the cursor down one line while remaining in the same column.

The left arrow (←) key moves the cursor to the left one character position (column).

The right arrow (→) key moves the cursor to the right one character position.

The Main Keypad

The main keypad contains the standard alphabetic and numeric keys, punctuation marks, and special characters found on a typewriter-style keypad. In addition, there may be some of the following keys, depending on what type of terminal you have, that perform special functions.

Repeat

Using the repeat (REPT) key in conjunction with another key will keep inputting the other character for as long as you hold the two keys down. Use REPT carefully — it repeats very fast. (This key is on the cursor control keypad on D2 and D200 model terminals.)

BREAK

The BREAK key puts your terminal back in text mode. If, for instance, you mistakenly type the contents of an unprintable program file, you might put your terminal into binary mode. In this mode, your terminal will not accept a CTRL-C CTRL-A sequence to interrupt the console. To put your terminal back in text mode, press the break key followed by CTRL-S, CTRL-C, CTRL-A, and CTRL-Q. For more on these keys, see the section on the Control key later in this chapter. (This key is on the cursor control keypad on D2 and D200 terminals.)

Delete

The delete key (DEL), named RUBOUT on some keyboards, will erase characters after you have typed them but before you enter them. On video display terminals, DEL will erase the character immediately preceding the cursor, and move the cursor to the erased position. On hard-copy terminals where you cannot move the cursor backwards, DEL will echo a backarrow or underscore. The system ignores DELs if there are no characters on the current input line. (Some terminals will return a bell if you try to delete a nonexistent character.)

NEW LINE and Carriage Return (CR)

The NEW LINE and CR keys are terminators. If you are typing a command, hitting NEW LINE or CR tells the CLI to begin executing the command. If you are typing something into a file, NEW LINE or CR moves the cursor to the leftmost position of the next line. The only difference between these two terminators is that CR erases characters to the right of the cursor while NEW LINE does not.

SHIFT

The SHIFT key creates alphabetic characters in uppercase and creates the shift characters for all other keys.

ALPHA LOCK

The ALPHA LOCK key is a toggle key that alternates between two functions. If your alphabetic characters are all lowercase, pressing the ALPHA LOCK key causes all subsequent alphabetic characters to be uppercase. Conversely, if all your characters are uppercase, pressing the ALPHA LOCK key causes subsequent alphabetic characters to be lowercase.

Control

The control (CTRL) key is like a second shift key. It permits a key to have a third value. To enter a control character, press and hold the CTRL key and then press another key. Throughout the manual, we denote a control character as

CTRL-char; e.g., CTRL-C

There isn't any difference between uppercase and lowercase control characters. Unlike other characters, control characters can interrupt a program or the operation of your terminal.

The CLI will accept and execute control characters even if it is in the process of executing some other CLI command.

Tables 1-1, 1-2, and 1-3 list the control characters that the CLI recognizes.

Table 1-1. Control Characters

Control Character	Effect
CTRL-C	Begin a control character sequence
CTRL-D	Indicates an end-of-file
CTRL-O	Discard or restart data written to the terminal
CTRL-P	Transmit the next character literally without interpreting it
CTRL-Q	Resume display of data on the terminal
CTRL-S	Freeze terminal display
CTRL-T	Reserved
CTRL-U	Cancel current input line
CTRL-V	Reserved

CTRL-C signals the start of a control character sequence. System action depends on the next character you type. Typing CTRL-C CTRL-B, for instance, will terminate the process while CTRL-C CTRL-C will merely echo the two CTRL-Cs and if you had typed ahead it empties the input buffer. Refer to the Control Character Sequences later in this chapter.

CTRL-D indicates an end-of-file. You can use it to terminate input from the terminal to the CLI.

NOTE: Two CTRL-Ds typed in a row will terminate the CLI which logs you off of the system.

CTRL-O discards information being output by an executing program rather than writing it to the terminal. If a task is writing data to the terminal when you type CTRL-O, it may actually execute faster since it no longer has to wait for the terminal to complete relatively slow data transmission.

The system cancels CTRL-O, sends output to the terminal screen, when it receives another CTRL-O or when something issues a forced write to the console. For instance, the TRACE command issues forced writes, so you cannot change the page mode displays of tracing. In other words, you *must* issue a CTRL-Q for every page. (Turning off the console will not suppress the output in this case either.) (Note that CTRL-Q does not restart output discarded by CTRL-O.) If a terminal has gone into binary mode, it may not be possible to tell if CTRL-O is on or off. In such a case, striking the BREAK key causes the PMGR program (Peripheral Manager) to clear CTRL-O at the same time it clears binary mode.

CTRL-P is the *literal* character. It tells the system not to interpret the next character you type. This allows you to input to a user program any of the following control characters: CTRL-C, CTRL-O, CTRL-P, CTRL-Q, CTRL-S, CTRL-T, and CTRL-V. If you type two consecutive CTRL-Ps, the system interprets the first as a *literal* character and passes the second to the task that performed the read operation.

CTRL-S stops displaying information on the terminal and suspends the task that is writing to the terminal. (If the current process has only one task, CTRL-S will block it when it tries to write to the terminal.) You can cancel CTRL-S and resume displaying information by typing CTRL-Q. The system automatically cancels CTRL-S when the process terminates.

CTRL-U cancels the current input line. Generally the current input line consists of all the characters you've typed since the previous NEW LINE. On video display terminals, CTRL-U erases a single input line. On hard-copy terminals, the system echoes U followed by a NEW LINE.

NOTE: If you have continued a CLI command over more than one screen line, CTRL-U cancels only the current input line, not the entire command line. Use CTRL-C CTRL-A to cancel the entire continued command line.

Control Character Sequences

You can affect the execution of a process that controls your terminal by typing a control character sequence. The first character of the sequence is always CTRL-C. The second character can be either CTRL-A, CTRL-B, CTRL-C, or CTRL-E. (See Table 1-2.)

Table 1-2. Control Character Sequence

CTRL-C followed by	Effect
CTRL-A	Interrupt the process if it has defined a console-interrupt service task
CTRL-B	Terminate the process
CTRL-C	Echo ↑C↑C on the terminal and empty input buffer
CTRL-E	Terminate the process and create a break file

CTRL-C CTRL-A may interrupt the process that has control of the console. If the process has defined a console-interrupt service task, control transfers to that task.

Assembly language programmers should read the description of the ?INTWT system call in the appropriate programmer's manual (AOS or AOS/VS) to learn how to define console-interrupt service tasks in their user programs. The process must define a console-interrupt service task for the CTRL-C CTRL-A sequence to work.

You can also use CTRL-C CTRL-A to erase a CLI command line that extends to two or more lines.

CTRL-C CTRL-B aborts the process in control of the terminal. When you terminate a process, control goes to its father (see Glossary for a definition of a father process). If you type CTRL-C CTRL-B while in the CLI, the CLI itself terminates. If EXEC is the parent process, it logs you off the system. It is preferable to type the BYE command to log off of the system.

CTRL-C CTRL-C simply echoes ↑C↑C on the terminal and empties the input buffer if you typed ahead of the screen display.

CTRL-C CTRL-E terminates the process in control of the terminal and directs the system to create a break file that may be useful in program debugging. After the system creates the break file, control goes to the terminated process's father, as described under CTRL-C CTRL-B.

SCREENEDIT Control Characters

The SCREENEDIT Control Characters, like the cursor control keys, help you move the cursor around on the terminal screen. They also help you to insert characters, insert tabs, and delete characters. The characters listed in Table 1-3 will only work if SCREENEDIT is ON. (See Chapter 6 for more information about the CLI SCREENEDIT command.)

Table 1-3. SCREENEDIT Control Characters

Character	Effect
CTRL-A	Move to the end of the character string.
CTRL-B	Move to the end of the previous word.
CTRL-E	Enter/exit the insert character mode.
CTRL-F	Move to the beginning of the next word.
CTRL-H	Move to the beginning of the character string. (The HOME key on the function keypad has the same effect.)
CTRL-I	Insert a tab. (The TAB key on the function keypad has the same effect.)
CTRL-K	Erase everything to the right of the cursor. (The ERASE EOL key on the function keypad has the same effect.)
CTRL-L	Erase page and terminate command input.
CTRL-X	Move to the right one character. (The → key on the function keypad has the same effect.)
CTRL-Y	Move to the left one character. (The ← key on the function keypad has the same effect.)

The HELP Command

If you're ever in doubt about a CLI command, pseudo-macro, system utility, or general subject, you can ask the CLI to display an explanation of the topic. For example, to get information about CLI commands, simply type

```
) HELP *COMMANDS ↵
```

To get an explanation about a particular CLI command, enter

```
) HELP name ↵
```

where name is the CLI command you want explained. If you include the /V switch, you will get a more complete description of the subject. For example, to obtain information about the CLI DUMP command, type

```
) HELP/V DUMP ↵
```

The HELPV macro displays the more detailed HELP messages for the CLI commands, while putting the terminal into page mode. Page mode freezes the screen display and makes the messages easier to read. Page mode displays 24 lines of text on the screen at one time and then freezes the display. You press CTRL-Q to scroll the next 24 lines of text onto your screen. After you have displayed the last screenful of text on the screen, you press CTRL-Q to return to the CLI. For example, to display the detailed help message for the CLI COPY command, type

```
) HELPV COPY ↵
```

For more information about the HELP command and the HELPV macro, see Chapter 6.

End of Chapter

Chapter 2
The File System

This chapter explains the Advanced Operating System's (AOS and AOS/VS) file system, and tells you how to navigate through it with the CLI. We will describe files, filenames, directories, the directory tree, pathnames, templates, search lists, links, access control lists, generic filenames, and file types.

Files

A *file* is a collection of information. It may reside on many kinds of media, such as magnetic tape or punched cards; however, unless we specify otherwise, we are describing disk files.

You refer to files by their filenames. A filename can be a string of 1 to 31 characters. The following are valid filename characters:

A–Z	Uppercase alphabetic characters
a–z	Lowercase alphabetic characters
0–9	Numerics
$	Dollar sign
_	Underscore
?	Question mark
.	Period

The CLI sees no difference between upper- and lowercase alphabetic characters in filenames. For example, the filenames FILE1 and file1 are identical.

File Types

There are many different kind of files. The operating system defines 255 file types. (See Table 2-6 at the end of this chapter.)

Although there are many different types of files, in this section we place files into one of two categories: directory files or data files.

Directory Files

A *directory* is a file that catalogs and contains information. You use directories to access other files and devices. You can think of directories as vestibules in a library. Directories do not actually contain any usable information, but they do contain passageways to other directories and data files. You use directories to organize your data files, but you never store information in them.

Data Files

Data files, on the other hand, are like the books in a library. Data files contain information, but they do not allow you mobility. You cannot use a data file to get to another file.

Filename Extensions

Some system utilities assume that filenames have specific extensions. (You may hear filename extensions referred to as filename suffixes.) For example, if you want to assemble a file named PROG.SR, you need only supply the macroassembler utility with the name PROG and it will find PROG.SR (if it can't find PROG.SR it will search for PROG). The macroassembler then creates an object file with the name PROG.OB. Likewise, you can give the Link utility the filename PROG and it will link the file PROG.OB and create a program file PROG.PR.

Table 2-1 contains some of the extensions (suffixes) that AOS and AOS/VS system utilities recognize.

Table 2-1. Filename Extensions

Extension	What Type of File it Specifies
.BRK	Breakfiles (begin with ?)
.CLI	CLI macro files
.DG	DG/L source files
.FR	FORTRAN IV and FORTRAN 5 source files
.F77	FORTRAN 77 source files
.LB	Unshared libraries
.OB	Object files
.OL	Overlay files
.PL1	PL/1 source files
.PR	Program (executable) files
.RG	RPG II source files
.SL	Shared libraries
.SR	Assembly language source files
.ST	Symbol table files
.TMP	Temporary files (begin with ?)
.TXT	Text (data) files

Directory Files

There are two types of directory files, an ordinary and a Control Point Directory (CPD). A CPD has a set maximum size, while an ordinary directory's size limit is the total amount of space the user profile has. (See the SPACE command in Chapter 6 to see how to set or check on the amount of disk space a CPD has.)

The organization of all directories resembles a network like an inverted tree. We say that a directory is superior to the directories on the lower branches of the tree, with the directories on the lower branches subordinate to the directories higher in the tree.

Directories are valuable aids in keeping your files organized. Subordinate or subdirectories add yet another dimension to organizing your files.

Figure 2-1 shows a directory tree. The top directory in the tree is the root directory. Since you will be referring to it often, we gave it the filename : . Three of the root's subordinate directories are PER, UTIL, and UDD. Directory PER contains entries for each peripheral device and generic file (described later in this chapter); directory UTIL contains entries for each system utility; and directory UDD contains entries for each user directory.

NOTE: Do not confuse the root directory name with the colon that you will use to separate filenames in a pathname. A colon at the beginning of a pathname always represents the root directory, while a colon in the middle of a pathname acts as a separator between the filenames.

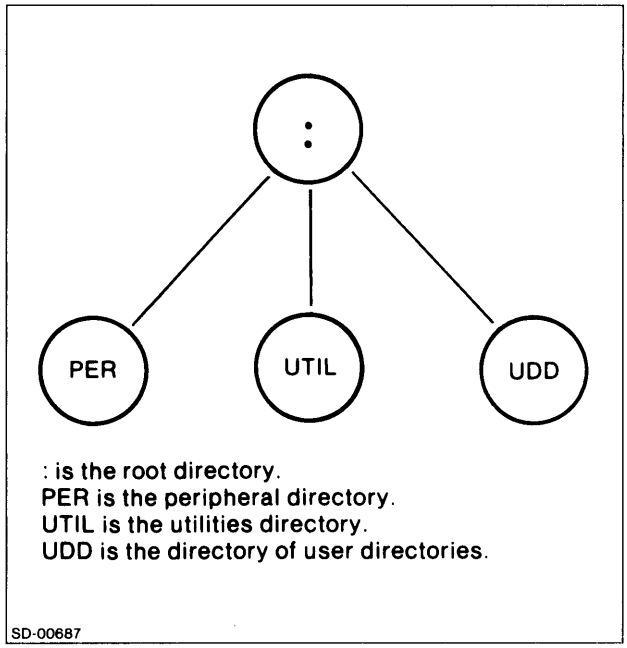


Figure 2-1. A Directory Tree

Every user starts with a directory subordinate to UDD. This *initial working directory* has the same filename as the user's username. For example, if your username is CHRIS your initial working directory is :UDD:CHRIS. As a system user, you may create additional directories below the initial working directory. To create a directory, use the CLI command CREATE with the /DIRECTORY switch (see the CREATE command in Chapter 6).

You can control how the system allocates disk space by designating certain directories in a Logical Disk (LD) as control point directories (CPDs). CPDs function exactly like other directories, except they restrict a file's disk space to a predefined limit. You specify this predefined limit with the /MAXSIZE= switch when you create the directory with the CREATE command. (For more information on CPDs, consult the *Programmer's Manual* for your system.)

If you want to create a directory file called MEMO, for example, enter the following:

```
) CREATE/DIRECTORY MEMO ⤸
```

Using the FILESTATUS command with the /ASSORTMENT switch, you can then check to see if you were successful. (See Chapter 6 for more on the FILESTATUS command.)

Enter the following line

```
) FILESTATUS/ASSORTMENT MEMO ⤸
```

The system should respond with something like

MEMO DIR 31-MAR-84 11:15:03 0

The DIR tells you that the file is a directory file. The other information tells you the creation date and time of the file and the file's size. (See the FILESTATUS command in Chapter 6.)

Figure 2-2 shows a directory tree with four user directories and several data files. Note that we represent directories as circles and data files as rectangles.

Your logical location within the directory tree is your reference point while you are on the system. We call this reference point your working directory. To refer to any file in your working directory, all you need is the file's name.

For example, in Figure 2-2 if your working directory is CAROL and you want to use the CLI command DELETE to delete FILE1, issue the command:

```
) DELETE FILE1 ⤸
```

Since FILE1 is in the working directory, the CLI immediately finds FILE1 and deletes it.

If, on the other hand, you want to delete a file that isn't in your working directory, you must tell the operating system where it can find the file. To do this, you must furnish a pathname to the file.

Pathnames

To refer to a file outside your working directory, you must use a *pathname*. Every file has a pathname. This pathname explains the path or route the system must take through the directory tree to find the file. A pathname can be a prefix alone, one or more filenames separated by colons, or a combination of prefixes and filenames. Pathname prefixes are symbolic representations of directory names. All filenames in a pathname must be directory names except for the last filename. The general format for a pathname is:

prefix *or* [prefix][directory-name:]... filename

Remember: Directories are themselves a type of file, so directory names are filenames too.

Table 2-2 contains the valid pathname prefixes.

Table 2-2. Pathname Prefixes

Prefix	Definition
: (colon)	Start at the root directory. This prefix forces your pathname to begin at the apex of the directory tree unlike other pathnames which begin at your working directory.
= (equal sign)	Start at the working directory. This is the default prefix. If you do not include a prefix in a pathname, the system will look first in the working directory and then in the search list directories. If you use the equal sign, the system will not look through your search list (see the section on search lists later in this chapter).
^ (uparrow, shift six)	Move up to the parent directory. This is the only way, other than specifying a superior directory name, to move up the directory tree. You can use more than one uparrow as the prefix for a pathname to move up more than one directory.
@ (at sign)	This starts you at the peripheral directory. This is equivalent to :PER:.

Every file has a pathname that starts with a colon (the root directory) and is a unique and a complete pathname for that file.

By using the equal sign or uparrow prefixes, you can move to a file in relation to your present directory. These pathnames explain to the system how to get to the file in relation to your working directory.

Except for the uparrow prefix, pathnames always move down the directory tree. If you have two filenames in a pathname, the second one must be immediately subordinate to the first. If, for example, your working directory is CAROL (see Figure 2-2), you cannot refer to FILE2 with the pathname =UDD:TED:FILE2. This is an illegal pathname since UDD is superior to the working directory, CAROL; the CLI will return an error. The pathname TED:FILE2, however, will refer to FILE2.

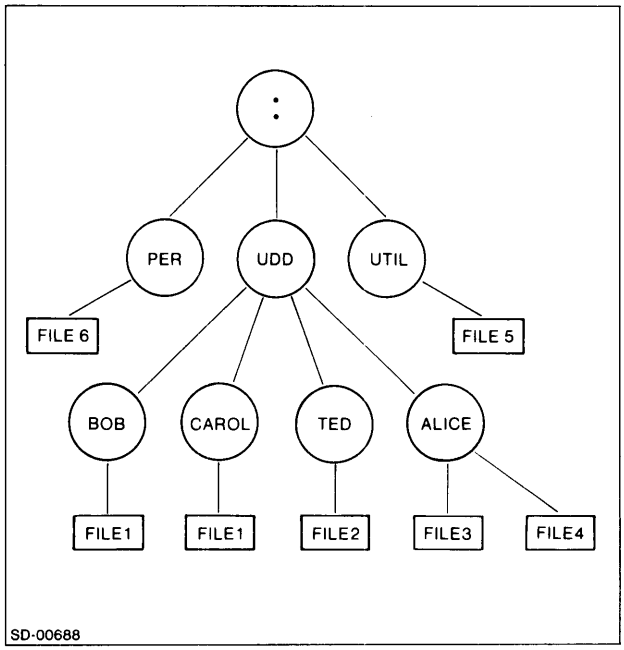


Figure 2-2. Sample Pathname

Pathname	Referenced file
FILE1	FILE1 (under CAROL)
↑BOB:FILE1	FILE1 (under BOB)
@FILE6	FILE6
:PER:FILE6	FILE6
↑↑PER:FILE6	FILE6
↑TED	TED
↑↑	: (root directory)
:	: (root directory)
:UTIL:FILE5	FILE5
=	CAROL

Figure 2-2 shows several pathnames and the path the system travels to find the files they specify.

When you supply a pathname in a CLI command, your working directory doesn't change.

You can always find the name of your working directory by entering the DIRECTORY command without any arguments. For example, if in Figure 2-2 the working directory is CAROL, the command

) QPRINT :UTIL:FILE5 ⤸

prints FILE5 in directory UTIL. If you enter

) DIRECTORY ⤸

the system will respond with

:UDD:CAROL

Your working directory is still :UDD:CAROL.

Templates

A template character is a special character that the CLI interprets symbolically rather than literally. Templates are valuable tools when you forget a file's name.

A pathname template is a pathname that contains template characters. In a sense, pathname prefixes are template characters since they are also symbolic characters. However, you can use prefixes only at the beginning of a pathname. The template characters that we describe in this section can occur anywhere within the pathname. Table 2-3 lists the five CLI templates and the type of filenames that they match. The number sign template helps you to refer to files within your working directory as well as any files in subordinate directories. The other templates only refer to files within your working directory.

Table 2-3. Template Characters

Template Character	What it Does
* (asterisk)	Matches any single filename character except a period
- (hyphen)	Matches any filename character string that does not contain a period, including a null string
+ (plus sign)	Matches any filename character string, including strings with a period and null strings
\ (backslash)	Tells the CLI to exclude any filename that matches the filename following this template
# (number sign)	Expands to + (any filename character string) and +:+ (any directory and any of its files) and +:+:+ (and so on until you match all subordinate directories and their files)

Asterisk

For example, suppose you were making changes to a file named TEST. The first time you made any changes to file TEST you created a new file and named it TEST1 . Then each time you updated the file, you copied the old file into a new file and added a 1 to the number extension of the filename. After awhile you might want to see all of the files of the different versions of file TEST.

By entering

)FILESTATUS TEST* ⤸

the CLI would respond with all of the files you named TEST 0–9 as well as just TEST.

If you made so many TEST files that your number extension was more than one digit (more than 9), you could enter

)FILESTATUS TEST** ⤸

to also retrieve any file that had a two-digit extension.

Hyphen

If you don't remember how many places your number extension expanded to, use the hyphen (–) template.

By entering

)FILESTATUS TEST- ⤸

the CLI will display all filenames that start with the first 4 letters of TEST, regardless of how many or what kind of characters follow it.

NOTE: The CLI is not case sensitive. In the previous example it would accept any combination of upper- and lowercase letters for the filename TEST. It would accept TEST or test and the response would be the same.

Plus Sign

The plus sign template character is useful if you want to refer to a file but you can't remember the exact filename. You are sure, however, that it begins with S .

If you enter

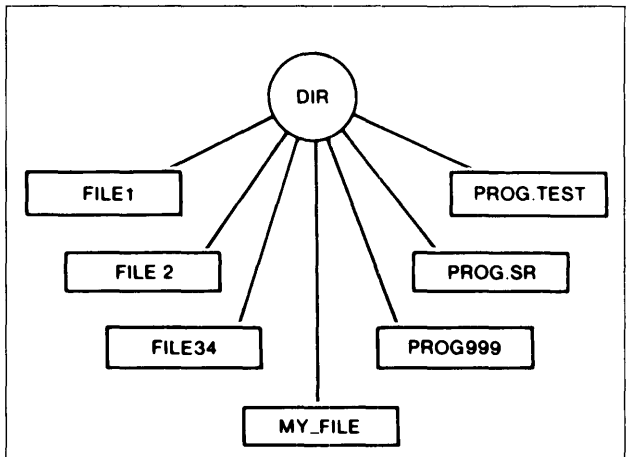
) FILESTATUS S+ ⤸

the CLI will return the names of all files in your current working directory that begin with S. This will narrow your search considerably.

Or perhaps you have a program named PROG1.PR which you want to move to another directory. But you also want to move all files associated with it (e.g., PROG1.OB, PROG1.SR). Rather than issuing a separate MOVE commands for each file, you could type

) MOVE PROGRAMS PROG1+ ⤸

and the CLI will move all files in your working directory that begin with PROG1 to the subordinate directory PROGRAMS. See Figure 2-3 for examples of template matching.



Template	Filenames Matched
FILE 1	FILE1
FILE*	FILE1 and FILE2
FILE-	FILE1, FILE2 and FILE34
PROG.-	PROG.SR and PROG.TEST
PROG+	PROG.SR, PROG999, and PROG.TEST
+	all filenames

SD-00689

Figure 2-3. Template Examples

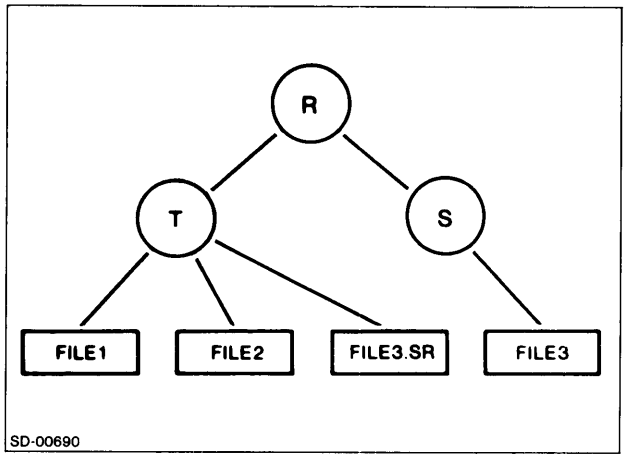


Figure 2-4. Directory Subtree

Referring to Figure 2-4, suppose R is your working directory. Observe how the CLI evaluates the following FILESTATUS arguments.

```
) FILESTATUS +:FILE* )
DIRECTORY :R:T
  FILE1        FILE2

DIRECTORY :R:S
  FILE3

) FILESTATUS T:+2+ )
DIRECTORY :R:T
  FILE2

) FILESTATUS +:+3- )
DIRECTORY :R:S
  FILE3

)
```

NOTE: If a pathname template matches a link file, it will not resolve the link. A link file is a file that contains a pathname to another file, usually in another directory. (See the section on links later on in this chapter.)

Number Sign

Unlike the other template characters, the number sign can expand to lengthy pathnames consisting of more than one filename. In other words, it can match files in subordinate directories. The number sign expands to the filename immediately preceding it, if any, and all its subordinate directories. If no pathname or filename template precedes the number sign, the system assumes the = prefix (the working directory). The pathname, :UDD:BOB:#, for instance, would expand to every directory in BOB's directory tree. You can think of # as a super plus sign, since

=#

is equivalent to the sum of

= (working directory)

and

=+:# (all data files and subordinate directory files under the working directory)

and

=+:+:+# (all data files under the working directory and all data files under each subordinate directory)

and so on. The expansion halts automatically when it matches the bottom of the tree, the most subordinate files.

This template character is very useful for finding a file when you don't know which directory it's in. Suppose, for example, that you have the directory structure shown in Figure 2-5. Your working directory is A and you know that you have a file called E but you don't know its exact location.

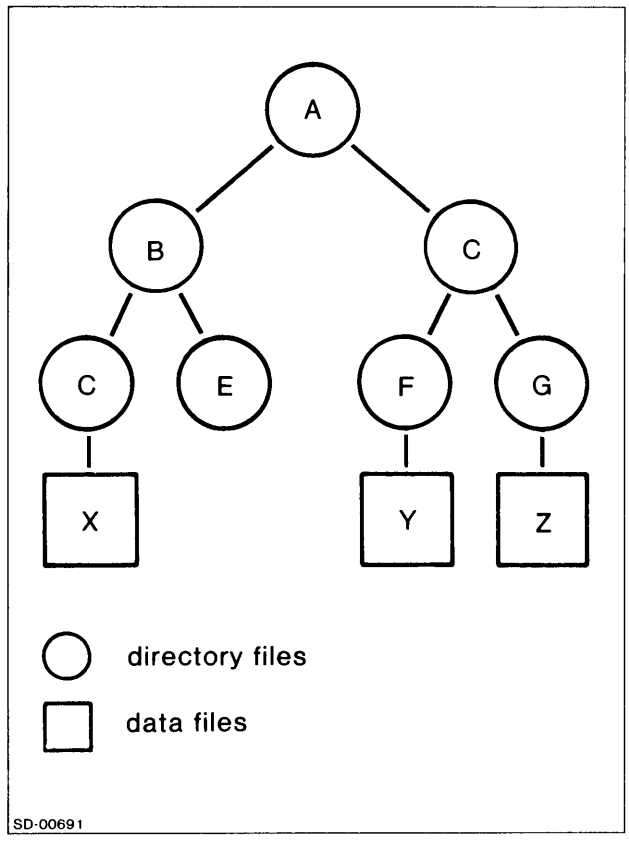


Figure 2-5. A Directory Subtree

If you type

) FILESTATUS E ↵

the CLI will return with nothing, not being able to find the file in the working directory. If however, you type

) FILESTATUS #:E ↵

The CLI will search through all subordinate directories and will output:

DIRECTORY :A:B
E
)

Note that you must include a colon after the number sign to separate the template # from the filename.

The number sign template character is also useful for accessing files with the same name but in different directories.

If you type

) FILESTATUS #:C ↵

the CLI will find both files named C and return:

DIRECTORY :A
C
DIRECTORY :A:B
C
)

Backslash

You use the backslash to restrict the set of filenames that match a filename template. The CLI will refer to every file that matches the filename template preceding the backslash, but will not refer to any files that match the filename template *following* the backslash. For example, the template

FILE+\FILE1+

matches every filename in the working directory that begins with FILE, except those whose names begin with FILE1. The pathname template

+\FILE2\+.SR

matches every filename in the working directory except FILE2 and those files ending in .SR.

In the subtree shown in Figure 2-6, suppose that directory R is the working directory. Consider the pathname template #\T. This template is equivalent to all of the following:

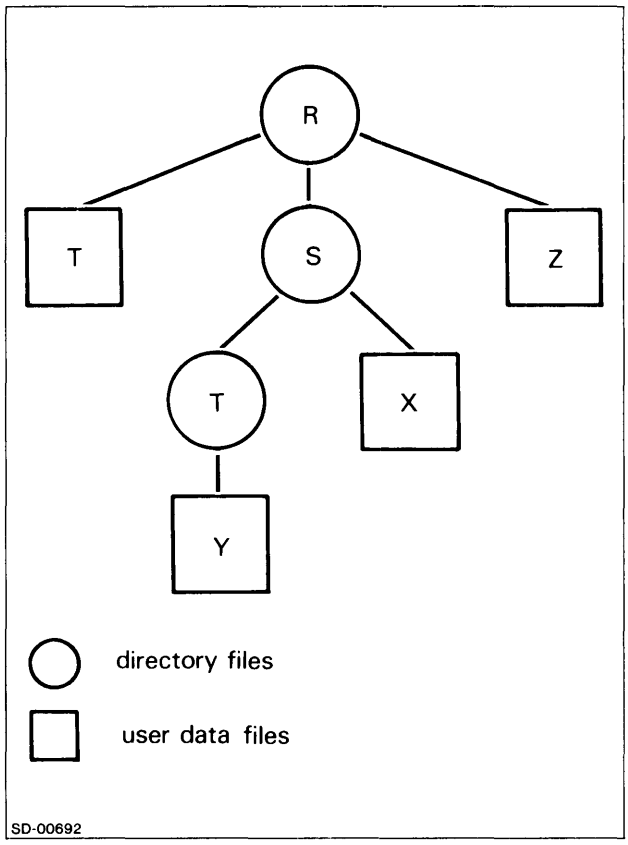


Figure 2-6. Another Directory Subtree

=#\T	(All files under the working directory except any named T.)
=+\T:#\T	(All files under the working directory and any subordinate directories and their files, except any named T.)
=+\T:+\T:#\T	(and so on..)

It matches the filenames R, S, X, and Z. Two files named T, one a data file and the other a directory, did not match. Note that since directory T does not match the template, the CLI doesn't search its subordinate files.

Colons separate each template in a pathname, and the backslash modifies only the template preceding it. Referring to Figure 2-5, the pathname

A:#\C

would evaluate to only one pathname (A:B:E) since the backslash excludes file C. However,the pathname

A:+\C:E

would match A:B:E since the backslash refers only to file C and not to C:E. Note that the colons define three groups (A: +\C :E).

Examples

The following examples refer to Figure 2-6. We assume that R is the working directory.

```
) FILESTATUS +:+\T ↵
DIRECTORY :R:S
  X

) FILESTATUS #\T ↵
DIRECTORY :R
  Z        S

DIRECTORY :R:S
  X

) FILESTATUS S:# ↵
DIRECTORY :R
  S

DIRECTORY :R:S
  T        X

DIRECTORY :R:S:T
  Y

) FILESTATUS #:T\S:T ↵
)
```

Note that in the last example the backslash referred only to S and not to S:T. So the CLI looked for a file whose pathname ended in T:T.

Search Lists

Quite often you use programs that aren't in your working directory; e.g., an editor, a compiler, or an assembler. Generally, the system stores these programs in directory UTIL. Assume that your working directory is DICK in Figure 2-7. If you want to assemble a file containing assembly code called FILE1.SR, you can type

```
) XEQ :UTIL:MASM FILE1 ↵
```

XEQ is the CLI command you use to execute a utility. The pathname :UTIL:MASM tells the system where to find MASM (the macro-assembler). This requires a lot of typing each time you want to assemble a file.

To keep the process of referring to files simple, the system provides a facility called a *search list*. The search list is a list of directories the system automatically searches when it can't find a file in your current working directory. You can display or change your search list with the SEARCHLIST command (see Chapter 6 for more information about this command). Using Figure 2-7, Suppose your search list is :UTIL,:PER, : and your working directory is DICK.

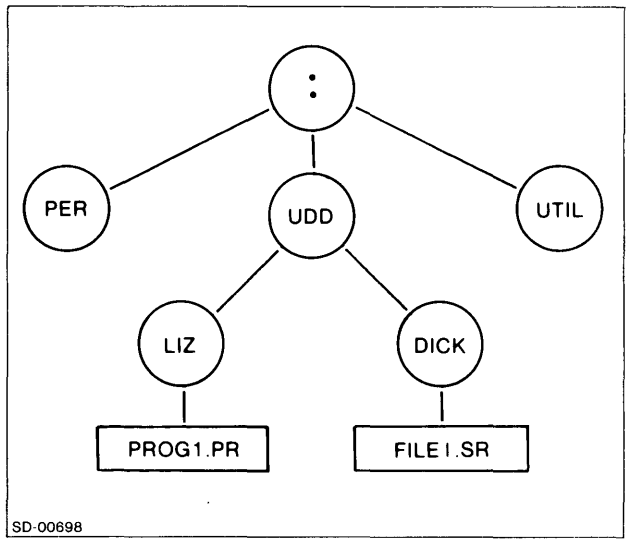


Figure 2-7. Search List Example

When you type

```
) XEQ MASM FILE1 ↵
```

the system finds MASM in UTIL and FILE1 in your working directory.

In case you have a file with the same name as a system file, you can prevent the system from searching the directories in your search list by using pathname prefixes. The system will not scan the search list if the pathname has a prefix. For instance, by using the = prefix, you can restrict the filename search to your working directory. You might want to do this, for example, to check on a file you created in your working directory. Suppose you created a file named UTIL. To check to see if you have a file named UTIL under your working directory enter

```
)FILESTATUS =UTIL ↵
```

and the system will restrict its search to your working directory.

Links

Another facility that simplifies referring to files is the link file. A *link* file is a file that contains a pathname or a pathname segment. When a link file's name appears in a pathname, the system replaces the link file's name with the contents of the link file, (the pathname it contains). Descriptions of exceptions to this replacement rule appear below.

You create link files with the CREATE/LINK command, and two arguments: the linkname and the pathname that you want to place in the file. (See the CREATE command in Chapter 6.)

In Figure 2-8, suppose directory A is the working directory. You create link B with the command

```
) CREATE/LINK B C:D )
```

B now contains the pathname C:D. The system will resolve the pathname B:E to C:D:E.

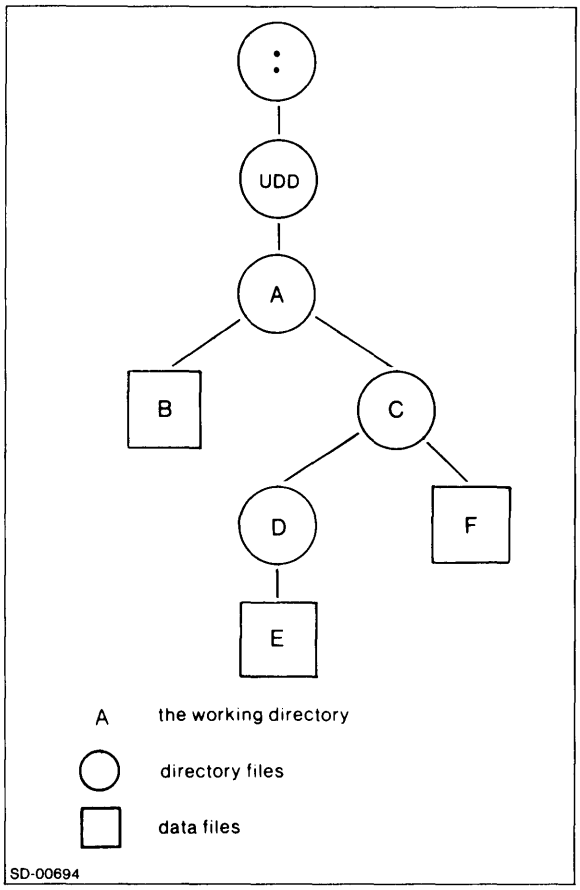


Figure 2-8. Link Examples

When the pathname within the link file begins with a pathname prefix, the CLI resolves the linkfile in a special way. The system uses the directory names preceding the link filename to locate the link file. It then discards these directory names and starts at the file specified by the link file's content's prefix. Assume that file F in Figure 2-8 is a link containing the pathname :UDD:A:C:D. Your working directory is A. If you use the pathname C:F:E, the contents of the link file does not simply replace the linkname, since the resulting pathname C::UDD:A:C:D:E would be incorrect. Instead, the pathname prefix : in the link file tells the CLI to start searching for the file over again, beginning at the system root (:). The correct resolution of C:F:E is :UDD:A:C:D:E.

This form of link resolution has special implications for the uparrow (↑) pathname prefix. If a link file contains a pathname beginning with an uparrow, its resolution is relative to the working directory and not to the directory where the link file resides. For example, assume that file F in Figure 2-8 is a link containing a single uparrow (↑). If your working directory is C and you give the simple pathname F, the system resolves the pathname to A. If your working directory is A, however,and you give the pathname C:F, the system resolves the pathname to UDD — that is, the directory superior to the working directory.

NOTE: If a pathname template matches a link file, it will not resolve the link.

When given a link filename as an argument, most CLI commands resolve the link. However, a few commands operate on the link file itself and not on the file it resolves to. Such commands include DELETE, DUMP, FILESTATUS, LOAD, and MOVE. For example, if DELETE has a link filename as an argument, it deletes the link file. Similarly, FILESTATUS displays the status of the link file, MOVE moves the link file, and DUMP and LOAD only dumps or loads the link file, and not the contents of the file that the link file contains.

Access Control List

Both AOS and AOS/VS allow you to restrict access to any file. You have control over how other people can use your file. You accomplish this by specifying an access control list (ACL) with the ACL command.

The access control list contains a list of users who can access the file, plus what each user can do with the file. The five types of file access follow:

- Execute access
- Read access
- Append access
- Write access
- Owner access

The effect of each access type depends on whether the file is a directory or nondirectory file. (See Table 2-4.)

CAUTION: If you give another user Owner access to a file, that user can change the ACL for the file.

Use the following format for ACLs:

username-template access-types
[username-template access-types...]

Table 2-4. Access Types

Access	Abbreviation	Non-Directory	Directory File
Execute	E	User can execute file	User can use directory's name in a pathname
Read	R	User can read (examine) data in files	User can examine list of files in directory
Append	A	N/A	User can insert names of new files into directory
Write	W	User can modify file's contents	User can insert and delete files from directory and change ACLs
Owner	O	User can change file's ACL and delete file	User can change directory's ACL or delete the directory

Don't forget to give yourself access at the same time. If your username is left off, you have absolutely no access to what was once your file. Each user has the type of access that follows the first template in the ACL line that matches his or her username. If there isn't any reference to the username or the username does not match any template in the file's ACL, then the user has no access to that file. For example, given the file with the following ACL:

```
JOHN+          E
+SMITH         RWE
JOHNSMITH      OWARE
```

the user JOHNADAMS has Execute access, and the user ADAMSMITH has Read, Write, and Execute access to this file. Note even though the third entry in the ACL gives JOHNSMITH all access types, he has only Execute access to the file since his username matches the first username template. For this reason, we usually order ACLs so that the earlier username templates match the fewest number of users, therefore giving the most access to the fewest number of users. The ACL for FILE1 would be better (from JOHNSMITH's point of view) if it were

```
JOHNSMITH      OWARE
+SMITH         RWE
JOHN+          E
```

Now JOHNSMITH has all access types, ADAMSMITH has Read, Write, and Execute access, and JOHNADAMS has Execute access only. Users whose usernames do not begin with the character string JOHN or end with the character string SMITH do not have any access to this file.

When you create a file or when a system utility creates a file for you, its ACL is your default ACL. (See the description of the CLI command DEFACL in Chapter 6.)

You use the ACL command to change and display the ACL of all files to which you have Owner access.

Generic Filenames

Generic filenames are filenames that simplify the use of certain common files and devices. By using a generic filename, a process need not name specific files or devices such as specific terminals. The system needs to know where to receive input and where to send output. Sources of input and recipients of output can differ depending on what you are asking the system to do.

The following are generic files:

```
@CONSOLE
@INPUT
@OUTPUT
@LIST
@DATA
@NULL
```

For interactive users, the @INPUT and @OUTPUT generic filenames are usually equivalent to @CONSOLE. In general, the CLI does not have an @DATA file. Interactive users usually do not have an @LIST file if their father is EXEC. For batch users, EXEC creates their @LIST disk file and names it

username.LIST.sequence-number

The disk file is created in the directory :QUEUE. @INPUT is a disk file; and @OUTPUT is the file you specify with the switch /QOUTPUT= when you queue a job. If you don't specify an output file, EXEC creates a default disk file in directory :QUEUE called username.OUTPUT.sequence-number. When the job completes, EXEC places this file in the BATCH_OUTPUT queue which can then be printed.

In addition, there is an @NULL file. Unlike the other generic filenames, @NULL does not correspond to any actual filename. Instead, any output to @NULL is thrown away, and a read from @NULL generates an end-of-file condition.

Device Names and Queue Names

Directory PER lists all system device names and queue names as well as the generic filenames. When you refer to them, you must always use the pathname prefix @. Table 2-5 contains the pathnames that are in directory PER.

Since a magnetic tape may contain one or more files, you must specify a file number on the tape. Use a colon to separate the tape unit name and the file number. For example, @MTB0:0 names the first file on magnetic tape unit 0, and @MTB0:1 names the second file on magnetic tape unit 0. If you omit a number, the default tape file number is 0.

You may create a file with any device or queue name within and beneath your user directory. The system will not confuse your filename with the device or queue name because your file does not have the @ prefix and will not be in directory :PER.

Table 2-5. Device and Queue Names

Device or Queue Name	What It Is
@DPxn, @DPx1n	First or second moving-head disk controllers. Letter x indicates type; number n indicates unit number. See the proper appendix of the AOS or AOS/VS operator's guide for details.
@DKB, @DKB1	Fixed-head disk drives
@LFD	(AOS/VS only) Labeled floppy diskette
@LPT, @LPT1	The first and second line printer (queue names)
@LMT	Labeled magnetic tape unit
@MTA, @MTA1	(AOS only) Magnetic tape controller 0 or 1 (device names)
@MTB, @MTB1	Magnetic tape controller 0 or 1 (device names)
@MTC, @MTC1	Magnetic tape controller 0 or 1 (device names)
@MTD, @MTD1	Magnetic tape controller 0 or 1 (device names)
@CON1, @CON2, ... @CONn	Console 1, 2, ... through n (device names)
@TRA, @TRA1	(AOS only) First and second paper tape readers (device names)
@PTP, @PTP1	(AOS only) First and second paper tape punches (queue names)
@CRA, @CRA1	First and second card readers (device names)
@PLT, @PLT1	First and second plotters (queue names)
@VCONn	Virtual consoles for networking

Labeled Magnetic Tapes

A magnetic tape is either *labeled* or *unlabeled.* An unlabeled tape has no tape file information recorded — it simply has tape files, whose names are numbers (0, 1, 2, 3, etc.). A labeled tape starts with a label which contains volume ID and file information. You refer to the tape by volume ID and tape filename.

Several different programs are involved with labeled tape access. The CLI commands MOUNT and DISMOUNT ask the system for a physical mount or or dismount of one or more labeled tape volumes. The EXEC program actually prompts the system operator to mount and dismount the tape volumes. Also the system checks that the tape volume IDs are correct. Finally, the LABEL program offers an easy way to label tapes.

Although labeling tapes requires extra effort, labeled tapes offer many advantages. These include

- You can store information on multiple tape volumes, without worrying about physical ends of tape. One tape file can span many tape volumes. This is useful for system backup and backup of large database files.
- The tape itself contains information about the material stored on it: user label text, filename, expiration date, and system ID.
- You control the amount of time each tape file will be retained. The system will not overwrite a labeled tape file until the retention period has passed or until the tape is relabeled.
- DG data management programs (INFOS® II and DG/DBMS) have logging utilities that expect labeled tapes.
- You can create tapes to be read on other operating systems, like an IBM system or a system that uses ANSI-standard labeled tapes.

You can label tapes with the LABEL utility, described in Chapter 6. For details on labeled tapes, consult Chapter 7 or the appropriate *How to Generate and Run* manual for your operating system.

Labeled Diskettes

Labeled diskettes, like labeled tapes, allow material to be written to multiple volumes. Labeled diskettes are needed for backup on systems that don't have a tape unit, but do have a diskette unit, or in any situation where someone wants to write to diskette and the material involved exceeds the capcaity of one diskette.

Using labeled diskettes is somewhat simpler than using labeled tapes, since the CLI can label diskettes and keep track of diskette volumes. (EXEC is not involved in and does not support labeled diskette operations.)

To enable the CLI to access labeled diskettes, use the OPERATOR command, described in Chapter 6.

For details on labeled diskettes, consult Chapter 7 or the appropriate *How to Generate and Run* manual for your operating system.

File Types

The operating system stores practically all information in files. It uses file types to distinguish files used for different purposes. The operating system provides 256 types of files. Table 2-6 lists the file types 0 through 127 that AOS and AOS/VS define. Your installation can define 128 additional file types (numbers 128 through 255), with /TYPE arguments to the CREATE, DUMP, FILESTATUS, LOAD, and MOVE commands (Chapter 6). For some files types, you can define the type of file when you create it by using the /TYPE=(type) switch. (See the CREATE command in Chapter 6.) You can find out what type file you have by adding the /ASSORTMENT switch to the filestatus command. (See Chapter 6.)

Table 2-6. File Types

Number	Mnemonic	Type	Number	Mnemonic	Type
0	LNK	Link	43–48	---	Reserved
1	SDF	System data file	49	CON	Console (hard copy, CRT, or virtual console)
2	MTF	Magnetic tape file	50–59	---	Reserved
3	GFN	Generic filename	60	SYN	Synchronous communication line
4–9	---	Reserved	61	---	Reserved
10	DIR	Disk directory	62	LUG	System Network Architecture Logical Unit Group
11	LDU	Logical disk	63	---	Reserved
12	CPD	Control point directory	64	UDF	User data file
13–19	---	Reserved	65	PRG	AOS program file
20	DKU	Disk unit	66	UPF	User profile file
21	MCU	Multiprocessor communications unit	67	STF	Symbol table file
22	MTU	Magnetic tape unit	68	TXT	Text file
23	LPU	Data channel line printer	69	LOG	System log file (accounting file)
24	LPD	Data channel line printer 2	70	NCC	FORTRAN carriage control file
25–29	---	Reserved	71	LCC	FORTRAN carriage control file
30	IPC	IPC port entry	72	FCC	FORTRAN carriage control file
31	---	Reserved	73	OCC	FORTRAN carriage control file
32	SPR	Spoolable peripheral directory	74	PRV	AOS/VS program file
33	QUE	Queue entry	75	WRD	Word processing file
34	LMT	Labeled tape	76	AFI	APL file (AOS/VS only)
35	LFD	Labeled floppy diskette	77	AWS	APL workspace file (AOS/VS only)
36	TRA	Paper tape reader	78	BCI	Basic core image
37	CRA	Card reader	79–87	---	Reserved
38	---	Reserved	88	BBS	Business BASIC save file
39	TPA	Paper tape punch	89	VLF	Business BASIC volume label file
40	PLA	Plotter	90	DBF	Business BASIC data base file
41	LPA	Programmed I/O line printer	91–127	---	Reserved
42	LPC	Programmed I/O line printer 2	128–255	---	User-defined file types

End of Chapter

Chapter 3
Command Line Syntax

This chapter describes the syntax of CLI command lines. It explains how to use command and argument switches and how to use parentheses and angle brackets to repeat commands. This chapter also describes command abbreviations, multiple command lines, and continuation lines.

CLI command lines have the form:

COMMAND*[switches] [argument[switches]...]...*

That is, a command line may consist of a command alone or a command and one or more arguments. An argument can be a filename, a process name, a device name, etc. Some commands require arguments, others allow arguments, and still others do not accept arguments. If a command line contains one or more arguments, you must separate the command and each argument with a separator. The following characters or character combinations serve as *separators* in CLI command lines:

- one or more spaces (except at the beginning or end of a command line)
- one or more tabs (except at the beginning or end of a command line)
- a single comma
- any combination of spaces, tabs, and a single comma

When the CLI formats a command line, it converts each separator into a single comma.

You can specify a null argument by typing two consecutive commas or a separator followed by a delimiter. (You might occasionally use null arguments, especially in command lines containing macros or pseudo-macros.) The valid CLI *delimiters* are NEW LINE, null, form feed (CTRL-L), carriage return (CR), and end-of-file (CTRL-D). On non-ANSI standard terminals, use carriage return for NEW LINE.

Switches

Append one or more switches to commands and arguments in order to select processing options and modify the execution of the command. Command switches modify commands. Argument switches modify arguments. All switches begin with a slash (/) and take one of two forms: simple or keyword. A simple switch has the form:

/switch

and a keyword switch has the form:

/keyword=value

For example, you can use the /L command switch to change the file to which the CLI writes output. Table 3-1 contains examples of the TYPE command, with and without the /L switch. The examples assume that you want to type the contents of MYFILE.

Table 3-1. Command Switches

Command	Switch	Effect
TYPE/L MYFILE	/L	Types output to the current LISTFILE.
TYPE/L=pathname	/L=pathname	Types output to the MYFILE file specified by pathname.
TYPE MYFILE	omitted	By default, types output to @OUTPUT. In interactive mode, @OUTPUT is usually the terminal.

Certain command switches require information pertaining to date and/or time. The /BEFORE and /AFTER switches of the LOAD and DUMP commands require the date and/or time. They allow you to do partial rather than full file backups. In such cases, you must input the date in the following format:

```
dd-mmm-yy
```

For example:

```
12-JUL-84
```

Notice you must use a 3-letter abbreviation (e.g., JUL) to identify the month, not a numeric value, and you must separate the parts to the date with a hyphen (–).

You must input the time in the format:

```
hh:mm:ss
```

For example:

```
01:30:00
```

for 1:30 a.m., or

```
13:30:00
```

for 1:30 p.m. The minutes and seconds are optional. The system enters 00 if you do not specify minutes or seconds. Therefore, entering 23 is equivalent to entering 23:00 or 23:00:00. All stand for 11:00 p.m. In addition, the system inserts 0 as the first member of the pair if you enter only a single digit for hours, minutes, or seconds. For this reason, entering 9:3 is equivalent to entering 09:03, that is, 9:03 a.m.

Coding Aids

There are two operators that the CLI provides to help extend the command lines. They are parentheses and angle brackets. Parentheses cause commands to repeat. Angle brackets expand the argument list.

Parentheses

If you enter a command followed by a group of arguments enclosed in parentheses, the CLI executes the command on all of the arguments one at a time. This saves you from having to enter the command line for each argument. One command line expands into many.

For example, the CLI command line

```
WRITE (A B C)
```

is equivalent to

```
WRITE A
WRITE B
WRITE C
```

If you enclose a subset of the entire argument list in parentheses, the CLI repeats the command for each argument in the subset, combining each expression in the parentheses with the other arguments on the command line.

For example:

```
WRITE A (B C) D
```

is equivalent to

```
WRITE A B D
WRITE A C D
```

If you place two or more commands within parentheses and follow them by an argument, the CLI executes each command in the list with the argument.

For example, the command line

```
(TYPE QPRINT) FILE1
```

is equivalent to the command lines

```
TYPE FILE1
QPRINT FILE1
```

If you enter a command line with two or more groups of arguments enclosed in parentheses, the CLI executes the command on the first argument in each group. It then executes the command on the second argument in each group, and so on. When the CLI exhausts one group, it executes the command on any remaining arguments in the other groups. Command repetition ends when the CLI executes the last argument in the largest group. For example, the command line

```
WRITE (A B C) (X Y)
```

is equivalent to the command lines

```
WRITE A X
WRITE B Y
WRITE C
```

Nested Parentheses

Nesting one set of parentheses within another set isolates the nested inner set from the outermost set. That is, the CLI treats the innermost set of arguments in such an argument list as a group.

For example, the command line

```
WRITE (A (B C) D)
```

is equivalent to the command lines

```
WRITE A
WRITE B C
WRITE D
```

You can enter a command line containing parentheses nested two or more levels. The CLI alternates in interpreting the parentheses as repeating operators and as grouping operators. It interprets the outermost parentheses as a repeating operator, the second-level parentheses as a grouping operator, the third-level as a repeating operator, the fourth-level as a grouping operator, and so on. You can nest parentheses to any depth.

For example, the command line

```
WRITE (A (B (C D) E) F)
```

is equivalent to the command lines

```
WRITE A
WRITE BCE
WRITE BDE
WRITE F
```

Angle Brackets

Angle brackets can help you expand a single command line. By combining arguments that contain the same character or character combination, the CLI creates a list of arguments for the command to act on. The CLI forms arguments by joining each character in the angle brackets with the characters that appear immediately before the left angle bracket and immediately after the right angle bracket.

For example, the command line

```
QPRINT FILE<1,2,3>
```

is equivalent to the command line

```
QPRINT FILE1 FILE2 FILE3
```

and the command line

```
QPRINT PROG<1,2,3>.SR
```

is equivalent to

```
QPRINT PROG1.SR PROG2.SR PROG3.SR
```

Now consider the command line

```
WRITE <A B C>_<X Y>
```

which the CLI expands to the command line

```
WRITE A_X A_Y B_X B_Y C_X C_Y
```

If you enter a command line containing two or more bracketed element lists, the CLI forms arguments as follows: it forms the first argument by taking the first element of the first bracketed group and combining it with adjacent character(s), as described earlier. In our example, A is adjacent to _, so the CLI produces A_. It then adds X from the second group to produce A_X. Then the CLI combines A_ with Y, the second character in the second group. When the CLI exhausts the second group, the CLI goes to the second element from the first group and adds it to the adjacent character(s), as above. This second iteration continues until the CLI exhausts the the second group. The total number of arguments the CLI forms equals the number of elements in the first group multiplied by the number of elements in the second group. There is no limit to the number of element groups you can type in a command line.

Nested Angle Brackets

The CLI allows you to nest any combination of angle brackets and parentheses. The possible combinations are:

- nesting angle brackets within angle brackets,
- parentheses within angle brackets,
- angle brackets within parentheses,
- and parentheses within parentheses.

However, an order of evaluation does exist. The CLI processes angle brackets first. It first expands the argument list by processing angle brackets from left to right. Next, it processes nested angle brackets from innermost to outermost. And finally, when there are no remaining angle brackets in the command line, the CLI processes parentheses in pairs from left to right. To gain a clearer understanding of command line expansion, study the following examples.

```
) WRITE (1 2) (1 2) ↵
1  1
2  2
)
```

The CLI executes the WRITE command twice: once for the first argument in each group, and then for the second argument in each group.

```
) WRITE <1 2><1 2> ↵
11  12  21  22
)
```

The CLI first expands the argument list by processing the angle brackets. It then writes the expanded argument list on the terminal.

```
) WRITE (1 2)<1 2> ↵
11  12
21  22
)
```

The CLI first expands the argument list by processing the angle brackets, which results in the argument list (11 21) (12 22). The CLI then repeats the WRITE command for the two argument groups as in the first example above.

```
) WRITE(<1 2><1 2>)↵
11
12
21
22
)
```

The CLI first expands the argument list by processing the two pairs of angle brackets which results in the argument list (11 12 21 22). The CLI then repeats the WRITE command for each argument enclosed in parentheses.

Abbreviations

You can abbreviate CLI commands and command switches. The shortest acceptable abbreviation is the smallest number of characters, beginning with the first character, that uniquely identifies the command or switch.

For example, X and XE are valid abbreviations for the XEQ command since it is the only command that begins with the character X. DE, however, is not a valid command abbreviation because DEASSIGN, DEBUG, DELETE, and DEFACL all begin with DE. Therefore, their accepted abbreviations are DEA, DEB, DEL, and DEF, respectively.

Multiple-Command Input Lines

You can enter two or more logically separate command lines on a single input line by separating the commands with a semicolon.

For example,

```
) TIME ; DATE
13:58:29
22-SEP-84
)
```

The CLI processes a multiple command line from left to right. If you enter an invalid command in a multiple-command line or if the execution of a syntactically correct command causes an error or abort, then the CLI will not execute commands that appear after the offending command. For more on error handling, see Chapter 4.

Continuation Lines

You can continue a command line to another line on the screen by typing an ampersand (&) before entering NEW LINE. The CLI issues the prompt *&)* on each continuation line. There is no limit to the number of continuation lines that the CLI will accept.

NOTE: The ampersand is not a separator. You must precede the ampersand with a separator or begin the continuation line with a separator if your command line requires a separator.

For example, the command line to link a FORTRAN IV main program and several subroutine modules is:

```
) XEQ LINK/L=PROG.LS/O=PROG MAIN& ↵
&),SUBR1,SUBR2,SUBR3,FSYS.LB,FORT0& ↵
&).LB ↵
```

Note that on the second input line, the leading comma separates the arguments MAIN and SUBR1. However, a single argument spans the second and third line because there is no separator before & on line 2 or .LB on line 3.

The CLI will automatically continue command lines if you exceed the maximum length before hitting the NEW LINE key. When SCREENEDIT is ON, the maximum line length is 76 characters; when SCREENEDIT is OFF, the maximum line length is 128 characters. If you continue a line, you cannot delete the first part of it with the delete key (DEL) or CTRL-U. If you want to cancel the input, type CTRL-C CTRL-A to interrupt the command line.

End of Chapter

Chapter 4
CLI Environments and Exceptional Condition Handling

Both AOS and AOS/VS are multiprogramming systems. This means that they can run many programs simultaneously. Each progam has its own share of the system's resources and the operating system considers it a process. Each process has a number of parameters that make up its environment. You can alter these parameters to suit what you are doing. Also, within each process you have a number of potential levels, each with its own environment. You can alter each level's environment independent of other levels' environments. Using the CLI PUSH command you can pass to a new level. Using the CLI POP command brings you back to the previous level. At the end of this chapter, there are two examples showing how you can use CLI environment levels.

CLI Environment

Your CLI environment consists of the following parameters. You can set and display the status of each one with a CLI command.

LEVEL
SUPERUSER
SUPERPROCESS
SCREENEDIT
SQUEEZE
CLASS1
CLASS2
TRACE
VARIABLES
LISTFILE
DATAFILE
LOGFILE
DIRECTORY
SEARCHLIST
DEFACL
STRING
PROMPT
CHARACTERISTICS

When you initially log on, these parameters have default settings. You can easily check the settings if you are in the CLI by typing the command CURRENT, which displays each environment parameter setting for the level where you currently are. In this chapter, we describe each of these parameters, what they mean and how to change your current environment.

LEVEL

When you first enter the CLI, you are at the highest environment level, LEVEL 0. The level number signifies how many levels deep you are, so that the deeper you are, the higher your level number. To display the current environment level, use the CLI command LEVEL. To change levels, use the PUSH command to descend one level and the POP command to ascend one level. PUSH adds one to the LEVEL setting and copies the values of all other parameters for the next level. POP subtracts one from the LEVEL setting and restores the other parameters to their previous settings. You can display the current level's parameters with the CURRENT command and the previous level's parameters with the PREVIOUS command, except when you are at level 0. When at level 0, there is no previous level.

NOTE: For the following four exception conditions — CLASS1=IGNORE, CLASS1=WARNING, CLASS1=ERROR, and CLASS2=ERROR — the CLI will compare your starting level with your current level when you encounter an error. If they are different, the current level will be output. You cannot suppress this output.

SUPERUSER

As described in Chapter 2, each file has an access control list that determines which users can access the file and which type(s) of access they have. If SUPERUSER is OFF, your process is subject to ACLs. If SUPERUSER is ON, you have access to every file in the entire directory tree, without regard to ACLs.

You can set and display whether SUPERUSER is ON or OFF with the SUPERUSER command. Only privileged users may set SUPERUSER to ON.

SUPERPROCESS

If SUPERPROCESS is set to OFF, you can modify only the state of your process and its sons. If SUPERPROCESS is set to ON, you can modify the state of every process in the process tree. (See the PROCESS command in Chapter 6 for a description of the variables that compose a process's state.)

You can set and display whether SUPERPROCESS is ON or OFF with the SUPERPROCESS command. Only privileged users may set SUPERPROCESS to ON. Any user, however, can obtain information about all processes regardless of the SUPERPROCESS setting by using the WHO and RUNTIME commands (see Chapter 6).

SCREENEDIT

When SCREENEDIT is ON, you can use the cursor control characters to modify the current line. (For more information about cursor control characters see Chapter 1 and for more information about SCREENEDIT see Chapter 6.) If SCREENEDIT is OFF, the cursor control characters echo on your terminal but have no effect.

You set and display the current screenedit mode by using the CLI SCREENEDIT command (see Chapter 6).

SQUEEZE

For several commands, the CLI formats its output into an easily readable layout. For example, the FILESTATUS command normally displays a table of filenames. To create the table format, the CLI must insert blanks and tabs to align columns. From time to time, you may want to eliminate all excess blanks and tabs. You do this with the SQUEEZE command.

When SQUEEZE is ON, it compresses two or more spaces or tabs into a single blank. When SQUEEZE is OFF, as it normally is, it does not compress spaces and tabs.

You set and display whether squeeze is ON of OFF with the CLI command SQUEEZE (see Chapter 6). You can use the /Q command switch to override the current squeeze setting for the duration of one CLI command.

CLASS1

The next section of this chapter describes Class 1 under exceptional conditions.

CLASS2

The next section of this chapter describes Class 2 under exceptional conditions.

TRACE

When debugging a large macro, it is useful to enable tracing. While the macros, psuedo-macros and commands are processing, you can see their full expansion with tracing on. You set and display whether tracing is ON or OFF with the CLI TRACE command (see Chapter 6).

Variables

There are 10 double-precision decimal variables available. Double precision is in the range 0 to 4,294,967,295. The initial value of each variable is 0.

You set and display the current value of the variables with the 10 CLI commands VAR0 through VAR9. You can insert the value of the variables into a command line using the 10 CLI pseudo-macros !VAR0 through !VAR9 (see Chapter 6).

LISTFILE

LISTFILE contains the pathname of the current generic @LIST file. You can code programs to write to @LIST instead of to a specific output file. To do this set LISTFILE to any file you wish before you execute the program. For example, you can have your program write to @LPT for one run and @MTB0:0 for the next without altering the program in any way by merely changing the LISTFILE setting before running the program.

You set and display the current list file with the CLI LISTFILE command. You can insert the list file's pathname in a command line with the !LISTFILE pseudo-macro (see Chapter 6). The /L= command switch allows you to change LISTFILE for the execution of one command. Interactive users generally do not have a generic @LIST, therefore, they must use the LISTFILE command before using the /L switch with a command.

DATAFILE

DATAFILE contains the pathname of the current generic @DATA file. You can code programs to read from @DATA instead of from a specific input file. Later, when you run the program, you can set DATAFILE to any file you wish. For example, you can read from the disk file on one run and from a magnetic tape unit on another run.

You set and display the current data file with the CLI DATAFILE command. You can insert the data file's pathname in a command line with the !DATAFILE pseudo-macro (see Chapter 6).

Note that this macro begins with the PUSH command and ends with the POP command. This allows you to set the environment as required for the duration of the macro, and restore the original environment just before leaving the macro.

Line 2 sets SQUEEZE mode OFF; that is, the CLI does not compress output. Lines 3 and 4 set the CLASS1 and CLASS2 exceptional conditions to appropriate severity levels for the duration of this macro.

The argument to the LISTFILE command on line 5 and the DATAFILE command on line 6 are dummy arguments. When you use dummy arguments, you must enter arguments along with the macro name. (Chapter 5 explains dummy arguments.) The CLI replaces %1% and %2% with the first and second arguments respectively that you supplied when you called the macro. (See Chapter 5). For example, you might call this macro with the CLI command line:

```
) SAMPLE @LPT @MTB0:1 ↓
```

The line printer becomes the LISTFILE and magnetic tape unit 0 file 1 becomes the DATAFILE.

Line 7 sets the working directory to MDIR. Line 8 sets the search list to include the pathname to NDIR, followed by the contents of the search list.

Line 9 invokes the user program MYPROG1. The /S switch appended to the XEQ command sets the STRING buffer to null and writes MYPROG1's termination message to the STRING buffer instead of to @OUTPUT (your terminal). Suppose you've designed MYPROG1 to return either the message 2 or 3 upon completion. The macro uses this message in deciding whether to run MYPROG2 or MYPROG3 next.

Commands that begin with the character ! are pseudo-macros (see Chapter 6). After MYPROG1 terminates, if STRING equals 2, the CLI executes MYPROG2, and if STRING does not equal 2, it executes MYPROG3.

After either MYPROG2 or MYPROG3 executes, the system returns to the previous environment, and then exits from the macro. Before returning to interactive mode, the CLI restores the previous level's settings.

Example Two

Another example displaying the usefulness of CLI levels follows.

Suppose you are working in one directory subordinate to your user directory and you need to check on a file in another directory also subordinate to your user directory. By changing levels, you can move to that other directory, check on the file and then just POP back to the directory where you were working.

When you PUSH to a new level and then POP back to your original level, your environment and working directory return to what they had been. By using the CLI levels, you save yourself from having to use the DIRECTORY command with a long pathname to move back to your original working directory.

The terminal dialog that follows shows a way to change levels, check on a file, and then return to your original level to continue working.

```
) DIRECTORY ↓
:UDD:SUE:TEST
)LEVEL ↓
LEVEL 0
)PUSH ↓
)LEVEL ↓
LEVEL 1
)DIRECTORY :UDD:SUE:RESULTS ↓
)DIRECTORY ↓
:UDD:SUE:RESULTS
)FILESTATUS/S/AS HIGH+ ↓

DIRECTORY :UDD:SUE:RESULTS

HIGH.CLI      UDF   17-APR-84    9:46:08   13825
HIGHSCHOOL    UDF   18-APR-84    9:57:40   36778
HIGHSCORES    UDF   12-APR-84   15:17:22   12185

)POP ↓
)LEVEL ↓
LEVEL 0
)DIRECTORY ↓
:UDD:SUE:TEST
)
```

```
)DIRECTORY ↓
:UDD:SUE:TEST
)LEVEL ↓
LEVEL 0
```

These two commands show us that we are in the directory :UDD:SUE:TEST and in level 0.

```
)PUSH ↓
)LEVEL ↓
LEVEL 1
```

We have now pushed to another level, level 1.

```
)DIRECTORY :UDD:SUE:RESULTS ↓
)DIRECTORY ↓
:UDD:SUE:RESULTS
)FILESTATUS/S/AS HIGH+ ↓

DIRECTORY :UDD:SUE:RESULTS

HIGH.CLI      UDF  17-APR-84   9:46:08   13825
HIGHSCHOOL    UDF  18-APR-84   9:57:40   36778
HIGHSCORES    UDF  12-APR-84   15:17:22  12185
```

We have now entered the subordinate directory :UDD:SUE:RESULTS and checked on all files that begin with the four letters, HIGH.

```
)POP ↓
)LEVEL ↓
LEVEL 0
)DIRECTORY ↓
:UDD:SUE:TEST
)
```

Having checked out the needed information on our file, we enter the POP command and check to see that we are back in level 0. We now check to see that our working directory has indeed returned to what it had been when we began without our having to move through the directory tree structure to get there.

End of Chapter

Chapter 5
CLI Macros

A macro (short for macroinstruction) is one command that stands for a sequence of commands. Normally, you store a CLI macro in a file and execute it by typing the filename. The system recognizes any filename with the filename extension of .CLI as a macro.

Suppose, for example, that you have a file called TDP.CLI that contains the following commands:

```
TIME
DATE
WRITE YOUR PROCESS NUMBER IS [!PID]
```

If you type TDP and then NEW LINE, the CLI will execute all three commands and write the output to your terminal:

```
) TDP ↵
12:05:33
26-JAN-84
YOUR PROCESS NUMBER IS 15
)
```

Macros come in handy if there is a sequence of CLI commands that you execute frequently. You can save time by putting the commands in a file and then simply typing the filename when you want to execute them. In addition, the CLI has conditional pseudo-macros that allow you to execute different commands within the same macro depending on the values of variables. (See the section on Conditional Pseudo-Macros later on in this chapter.)

Macro Names

Any legal filename is a legal macro name. In general, you want to end your macro filenames with the .CLI extension (see Chapter 2 for information about filename extensions). When you type a macro name, the CLI looks for the file in your working directory first. If it doesn't find the file there, it looks through your search list directories.

When you type:

```
) TDP ↵
```

the CLI will first look for a CLI command called TDP. Since there isn't a CLI command by that name it then adds a .CLI extension and looks for a file whose name is TDP.CLI. (In other words, the CLI checks to see if the file is a macro). If the CLI doesn't find a file with that name, it will look for a filename exactly as you entered it without the system added .CLI extension (i.e., TDP).

For example, if what you entered has an extension of .CLI:

```
) TDP.CLI ↵
```

the CLI will execute the macro because after not finding a CLI command with that name and after not finding a file with the filename TDP.CLI.CLI, the CLI would then look for, and find, the file TDP.CLI. Therefore, you don't have to add the .CLI extension when naming macros since the CLI will check to see if it is a macro anyway. But it is helpful to be able to display your files and to tell right away which are macros.

For example entering

```
) FILESTATUS +.CLI ↵
```

displays all of your macro filenames with the .CLI extension.

Be careful that your macro names are not the same as CLI commands. The CLI will execute the CLI command and not the macro. But if you enter the macro name along with its .CLI extension, the CLI executes the macro file. Also, if you enclose the macro name in square brackets, the CLI will execute the macro and not the command. Whenever you enter a filename in square brackets, the CLI replaces the filename with the contents of the file.

For example, if the file NAMES contained the lines

```
FILE_A
FILE_B
FILE_C
```

and you entered

```
) TYPE NAMES ↵
```

the CLI will type

```
FILE_A
FILE_B
FILE_C
```

But if you entered the line

```
) TYPE [NAMES] ⤸
```

the CLI will type out the contents of each file named within file NAMES; i.e.,FILE_A, FILE_B, FILE_C.

Creating Macros

To create a macro, just enter commands into a file. The macro name is the filename. There are two ways to enter commands into a file. The first is to use a text editor; e.g., SED, SPEED, or LINEDIT. Simply enter the commands into the file in the order you want them executed.

The second way to create a macro is to use the CLI CREATE command with the /I switch. This will create a file with whatever name you specify, and open it for input. The *))* prompt tells you that the CLI is waiting for input into the file. When you type a command and hit NEW LINE, the CLI will return another *))* prompt. It will not execute the command, nor even check to see if the syntax is valid — it will merely store whatever you type in the created file. To signal that you have finished putting commands into the file, type another close parenthesis next to the double prompt and then hit the NEW LINE key.

Example

To create the TDP.CLI macro previously listed, you would type:

```
) CREATE/I TDP.CLI ⤸
))TIME ⤸
))DATE ⤸
))WRITE YOUR PROCESS NUMBER IS [!PID] ⤸
)))⤸
)
```

If you want to change an existing macro, you must use a text editor or delete the file and then recreate it. There is no way to change the contents of a file using CLI commands. You can, however, add to a file by using the COPY command with the /A switch, or the WRITE or TYPE command with the /L=macrofilename switch (see Chapter 6).

Dummy Arguments

Macros are very versatile and convenient ways of executing a routine sequence of CLI commands. But you might want to enter a sequence of CLI commands to operate on arguments that vary each time you evoke the macro. You can use dummy arguments to represent the variable arguments.

Macros use dummy arguments to represent actual arguments. By actual argument, we mean the string or number that a command within the macro actually operates on. In the command line

```
DELETE FILEA
```

FILEA is the actual argument. In the command line

```
DELETE %1%
```

however, %1% is a dummy argument. The actual argument FILEA gets passed to the DELETE command within the macro. The CLI will not actually try to delete a file called %1%. Instead, it will decide what file %1% represents and delete that file. You must place dummy arguments within percent signs. This is how the CLI distinguishes them from actual arguments. You can use dummy arguments only in macros. A dummy argument can represent one actual argument, a range of actual arguments, or a macro switch. Table 5-1 summarizes the range argument default values, Table 5-2 summarizes the switch dummy arguments, and Table 5-3 summarizes the format of the three types of arguments.

Single Dummy Arguments

The simplest type of dummy argument contains a single number enclosed by percent signs. The number represents the numerical position of the actual argument when you called the macro. For instance, %1% would represent the first argument, while %3% would represent the third argument. %0% always stands for the macro-name itself. Suppose you have the macro SWITCH.CLI containing the following command lines:

```
RENAME %1% TEMP
RENAME %2% %1%
RENAME TEMP %2%
```

If you type:

```
) SWITCH FILEA FILEB ⤸
```

the CLI will substitute FILEA for every occurrence of %1% and FILEB for every occurrence of %2%. So the CLI executes the following commands:

```
RENAME FILEA TEMP
RENAME FILEB FILEA
RENAME TEMP FILEB
```

The net result will be that the two files — FILEA and FILEB — will exchange their names. If you had called the macro with only one argument (i.e., SWITCH FILEA), then %2% gets replaced by a null. Consequently, you would have received a CLI error (*WRONG NUMBER OF ARGUMENTS*) when the CLI attempted to execute the second command in the macro.

Range Dummy Arguments

A single dummy argument can represent more than one actual argument. The general format for a range dummy argument is:

%m-n,i%

where

m represents the first argument

n represents the last argument

i represents the increment value.

%2-4,1%, for instance, would represent the second, third, and fourth arguments. %2-4,2%, on the other hand, would represent only the second and fourth arguments since the 2 in the i-position tells the CLI to increment by 2 (i.e., skip every other one). %1-10,3% would represent the first, fourth, seventh, and tenth arguments.

The variables — m, n, and i — all have default values that the CLI assigns if you don't specify a different value:

- if you omit m,the CLI assumes it is 1;
- if you omit n, the CLI assumes 32767;
- if you omit i, the CLI assumes it is 1. (See Table 5-1).

Hence, %1-4,1%, %1-4%, and %-4% are all equivalent. If you call the macro with only four arguments, then %-% would also expand to the first, second, third, and fourth arguments.

Table 5-1. Range Argument Default Values

Dummy Argument Format	Value(s) Omitted	Default Value
%m-n,i%	None	Expands to every ith argument from the mth to the nth
%-n,i%	m	Expands to every ith argument from the first to the nth
%m-,i%	n	Expands to every ith argument from the mth to the last
%-,i%	m,n	Expands to every ith argument from the first to the last
%m-n%	i	Expands to every argument from the mth to the nth
%-n%	m, i	Expands to every argument from the first to the nth
%m-%	n,i	Expands to every argument from the mth to the last
%-%	m,n,i	Expands to every argument from the first to the last

Example

Suppose you have the macro RECREATE.CLI:

```
DELETE (%-%)
CREATE (%-%)
```

If you type:

```
) RECREATE FILEA FILEB FILEC ↵
```

the CLI will execute:

```
DELETE (FILEA FILEB FILEC)
CREATE (FILEA FILEB FILEC)
```

The result will be three empty files.

Or suppose you have the macro RNAM.CLI that contains this command:

```
RENAME (%1-,2%) (%2-,2%)
```

This will rename all odd-numbered arguments to their even-numbered counterparts.

For instance, the command

```
) RNAM A B C D E F ↵
```

would expand to:

```
RENAME A B
RENAME C D
RENAME E F
```

Switch Dummy Arguments

The CLI macro facility can use slashes and backslashes to signify the inclusion or exclusion of switches in a dummy argument. The format for specifying a switch is:

%dummy_arg/switchname%

The format for excluding a switch is:

%dummy_arg\switchname%

If you type only a slash or backslash, without a switchname, then the CLI assumes that you are referring to all switches for the particular argument. Table 5-2 illustrates how this works. If a switch takes a value (e.g., /COPIES=5), the expansion of the dummy arguments automatically includes its value.

Table 5-2. Dummy Argument Switches

Switch Dummy Argument	What it Represents
%1/S%	In the expansion of the dummy argument, include the first argument's /S switch.
%2\L%	In the expansion of the dummy argument, include all of the second argument's switches except /L.
%0/%	In the expansion of the dummy argument, include all of the macro names's switches.
%1/W/ff/G%	In the expansion of the dummy argument, include the first argument's /W, /ff, and /G switches. (The CLI is not case sensitive, so it would match these switches even if you called them /w/FF/g.)
%1/S=%'	In the expansion of the dummy argument, expand to the /S= switch's value.

There is also a dummy format to extract only a switch value rather than the entire switch. The format is:

%dummy_arg/switchname=%

If EXAMP.CLI were a macro, and you typed EXAMP/T=5, then %0/T=% would expand to 5.

Example

The following example shows how switch dummy arguments expand. See the CALCULATOR macro at the end of this chapter for an illustration of the full capability of switch dummy arguments.

```
) CREATE/I DSWITCH.CLI ↓
))WRITE THE FIRST ARGUMENT'S SWITCHES& ↓
)) ARE: %1/% ↓
))WRITE THE SWITCHES TO THE MACRO NAME& ↓
)) ARE: %0/% ↓
))WRITE THE VALUE OF THE /L= SWITCH& ↓
)) IS: %0/L=% ↓
))WRITE THE MACRO WAS CALLED WITH THE& ↓
)) FOLLOWING SWITCHES ↓
))WRITE NOT INCLUDING S AND T: %0\S\T% ↓
))WRITE THE FIRST ARGUMENT WITHOUT ITS& ↓
)) SWITCHES IS: %1\% ↓
))) ↓
)
) DSWITCH/L=7/S/T FILEA/D/R ↓
THE FIRST ARGUMENT'S SWITCHES ARE: /D/R
THE SWITCHES TO THE MACRO NAME ARE: /L=7/S/T
THE VALUE OF THE /L= SWITCH IS: 7
THE MACRO WAS CALLED WITH THE FOLLOWING SWITCHES
NOT INCLUDING S AND T: /L=7
THE FIRST ARGUMENT WITHOUT ITS SWITCHES IS: FILEA
)
```

Table 5-3. Dummy Argument Formats

Dummy	Argument Expansion
%0%	Expands to macro name (with all its switches).
%n%	Expands to the nth argument (with all its switches) where n is the numerical position of the argument.
%m-n,i%	Expands to every ith argument in the range from m to n inclusive.
%n/%	Expands to the nth argument's switches.
%n\%	Expands to the nth argument without its switches.
%n/s%	Expands to the nth argument's /s switch. The expansion includes the value of the /s switch if it takes a value. /s is an actual switch, not a variable.
%n/s=%	Expands to the nth argument's /s= switch's value.
%n\s%	Expands to all of the nth argument's switches except not the /s switch.

Macro Syntax

In general, the syntax for macros is the same as the syntax for CLI commands. (See Chapter 3.) However, in some instances the syntax for macros is different.

Parentheses in Macro Calls

Parentheses in macro calls work just as they do in commands. That is, you can use parentheses to make a single dummy argument represent more than one actual argument.

For instance, suppose the following command lines made up the macro ASM.CLI:

```
XEQ MASM/L=%1%.LS %1%
QPRINT %1%.LS
```

If you call the macro with

```
) ASM (FILEA,FILEB,FILEC) ↓
```

the CLI will expand it as follows:

```
XEQ MASM/L=FILEA.LS FILEA
QPRINT FILEA.LS

XEQ MASM/L=FILEB.LS FILEB
QPRINT FILEB.LS

XEQ MASM/L=FILEC.LS FILEC
QPRINT FILEC.LS
```

Although the %1% represents only one argument, the parentheses tell the CLI to execute the macro three times, substituting a different argument each time.

Brackets

When you type a filename and surround it with brackets, the CLI substitutes the contents of the file for the bracketed filename. This is essentially what occurs automatically when you execute a macro.

When you type:

```
) TDP ↓
```

the CLI looks for the file TDP.CLI (or TDP if it does not find TDP.CLI), and tries to execute the contents of that file. Here you are implicitly calling the macro. The CLI would respond identically if you typed:

```
) [TDP] ↓
```

The only difference is that the brackets make the call explicit. The bracket syntax also makes it possible to execute macros whose names are also commands. If you have a macro named TI.CLI and try to execute it by typing

```
) TI ↓
```

the CLI will think you are executing the TIME command and will output the time of day. If, however, you type

```
) [TI] ↓
```

the CLI will know you are referring to a file and will execute TI.CLI.

With brackets, moreover, you can use the contents of a file as an argument to a macro or command rather than as the macro itself. Suppose the following line made up the file CONTENTS:

```
FILEA,FILEB,FILEC,FILED
```

If you type

```
) QPRINT ([CONTENTS]) ↓
```

the CLI will expand the line to

```
QPRINT (FILEA,FILEB,FILEC,FILED)
```

and will therefore output the four files to the line printer.

NOTE: To use the contents of a file as input to a command, you need to override the delimiter at the end of each line. Otherwise, the system will interpret the delimiter as the end of the command line, and will not include the rest of the lines in the file as arguments to the command. You can create the file with SPEED or some other utility that does not automatically put a carriage return at the end of each line. If you use the CREATE/I command to create the file, you can end each line with the continuation character (&), which will override the delimiter.

Conditional Pseudo-Macros

The CLI supports conditional pseudo-macros that perform the same function as IF/THEN/ELSE statements in other programming languages such as PL/I and FORTRAN. They allow you to execute different command paths depending on various conditions.

Certain pseudo-macros can begin a conditional block of text. Each pseudo-macro takes exactly two arguments. You should use commas to separate the arguments, or spaces if neither argument is null. To show a null argument, use two consecutive commas. Two of the pseudo-macros, [!EQUAL] and [!NEQUAL], accept any strings as valid arguments: numbers, literal words, macros, or pseudo-macros. The [!EQUAL] condition is true if the two arguments are equal. The [!NEQUAL] condition is true if the two arguments are not equal. The other pseudo-macros require arguments that are unsigned decimal integers. Each condition is true if the first argument has the specified relationship to the second argument:

Name	Condition Tested For
[!UEQ]	equal to
[!UGE]	greater than or equal to
[!UGT]	greater than
[!ULE]	less than or equal to
[!ULT]	less than
[!UNE]	not equal to

The following discussion focuses on conditional blocks beginning with [!EQUAL] and [!NEQUAL], because they take the most general arguments. The other pseudo-macros function just like these two, except that their arguments must be unsigned decimal integers.

Each conditional section of a macro must begin with one of these pseudo-macros. The section must end with the [!END] pseudo-macro. You may also include an [!ELSE] between the two. Normally, the CLI will execute commands in a macro sequentially. When it encounters a conditional pseudo-macro, however, it compares the two arguments and, depending on the results, follows one of the two code paths. If the condition is true, the CLI steps to the next command in sequence; if the condition is false, the CLI jumps to the corresponding [!ELSE] statement, or to the [!END] if there is no [!ELSE]. Each conditional pseudo-macro and [!ELSE] pseudo-macro must have a corresponding [!END] statement. Figure 5-1 illustrates the routing for conditional pseudo-macros.

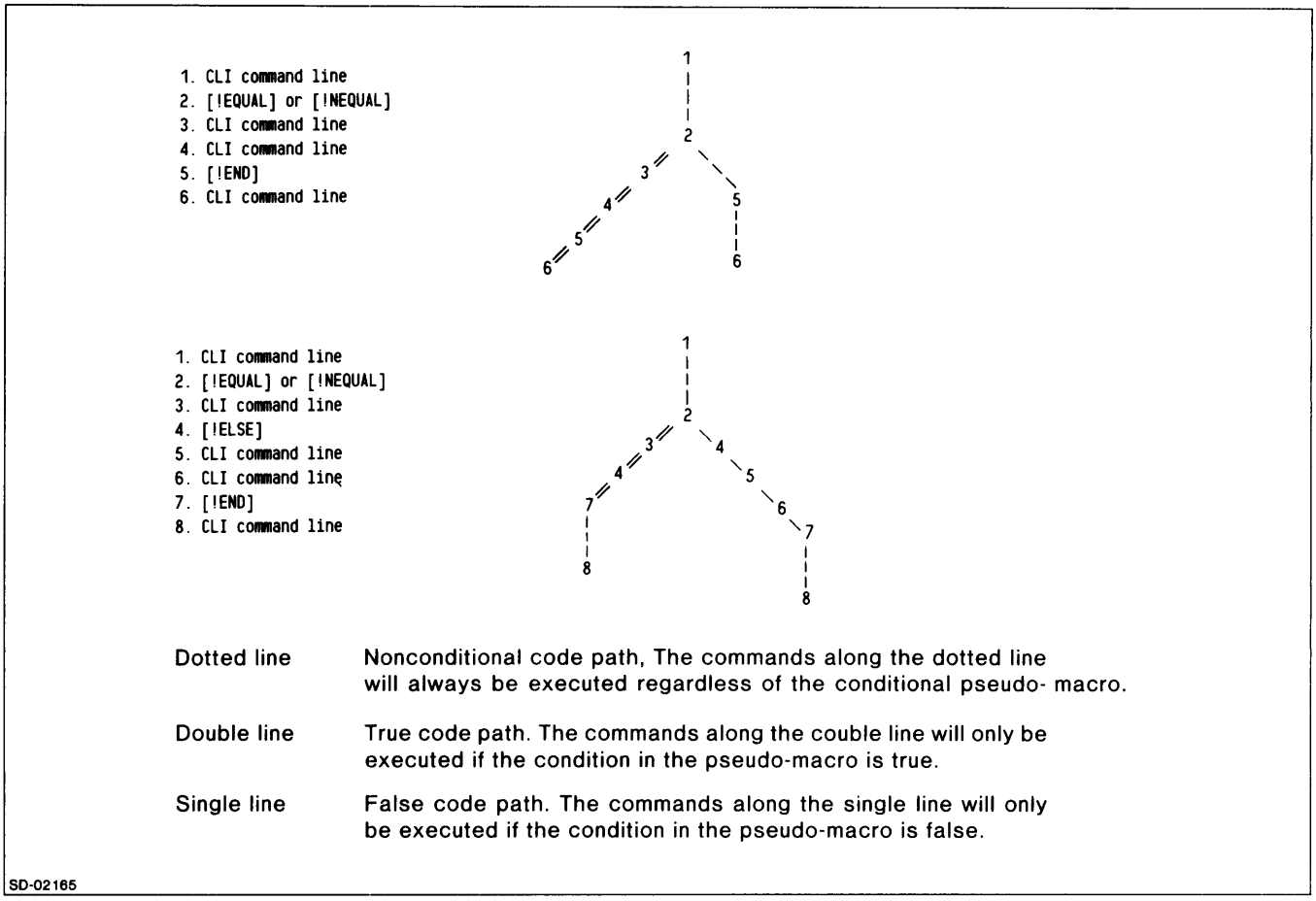


Figure 5-1. Conditional Code Paths

Examples

In the following example, we test whether the operator is on or off duty. If the operator is on duty, then we queue up the first argument. If the operator is off duty, the macro will print a message.

```
[!EQUAL,[!OPERATOR],ON]
    QBATCH %-%
[!ELSE]
    WRITE THE OPERATOR IS OFF DUTY.
[!END]
```

The first argument, [!OPERATOR], is a pseudo-macro that expands to either ON or OFF (see Chapter 6). The CLI compares the resulting expansion character by character with the second argument, ON. If they are equal, the CLI will QBATCH %-%; if they aren't equal, the CLI will print a message that the operator is off duty.

Note that you must include separators between the pseudo-macro and the first argument, and between the first and second arguments. The indentations on the second and fourth lines are cosmetic only — they make the macro easier to read, but the CLI ignores any spaces or tabs at the beginning of a line.

In the second example, the conditional branching pseudo-macro [!ULE] is used to emulate a DO increment loop or indexed loop. In PL/I, such a loop has the form DO I = r TO s BY t. This macro performs a specified action — a CLI command or macro, given as the first argument — a certain number of times. The second argument is the initial value, the third is the stopping value, and the fourth is the amount by which the second argument is incremented after each execution of the macro. The loop stops executing when the value of the second argument is greater than the value of the third argument.

The body of the macro, DO_INCREMENT.CLI, is as follows:

```
[!ULE %2% %3%]
WRITE CONTINUE LOOPING — INDEX &
□VARIABLE IS %2%
%1%
DO_INCREMENT %1% [!UADD %2% %4%] &
□%3% %4%]
[!ELSE]
WRITE STOP LOOPING — INDEX VARIABLE &
□IS %2%
[!END]
```

The macro is recursive, since it calls itself to perform each repetition of the loop. We included the two WRITE command lines simply for demonstration purposes.

We can demonstrate this macro by having it execute another simple macro, TEST.CLI, which contains a single command line: WRITE SIMULATE OPERATIONS. Invoking DO_INCREMENT causes these results:

```
) DO_INCREMENT TEST 2 10 3 ⌇
CONTINUE LOOPING—INDEX VARIABLE IS 2
SIMULATE OPERATIONS
CONTINUE LOOPING—INDEX VARIABLE IS 5
SIMULATE OPERATIONS
CONTINUE LOOPING—INDEX VARIABLE IS 8
SIMULATE OPERATIONS
STOP LOOPING—INDEX VARIABLE IS 11
)
```

Conditional Arguments

The CLI treats two arguments that you pass to [!EQUAL] and [!NEQUAL] as strings and compares them character by character. If one string is longer than another, they are unequal (e.g., [!EQUAL,FILEA,FILEAB] would execute to the false code path). The CLI does not distinguish between uppercase and lowercase characters.

Conditional arguments may be more complex than simple strings. For example, you might want to use a pseudo-macro as an argument. The statement, [!EQUAL,[!FILENAMES FILEA],] tells the CLI to execute the following command if FILEA does not exist. Note that the absence of a second argument is equivalent to a null. (See the [!FILENAMES] pseudo-macro in Chapter 6.)

A conditional argument may also be a conditional pseudo-macro. This means that the first and second arguments can vary depending on specified conditions. The one pseudo-macro statement:

```
[!EQUAL,[!EQUAL,%1%,%2%]YES&
  [!ELSE]NO[!END],&
[!EQUAL,%2%,%3%]YES&
  [!ELSE]NOT[!END]]
```

tests whether the first, second, and third arguments are all equal. The two arguments for the first [!EQUAL] are:

```
[!EQUAL,%1%,%2%]YES[!ELSE]NO[!END]
```

and

```
[!EQUAL,%2%,%3%]YES[!ELSE]NOT[!END]
```

There are four possibilities for the resolution of this pseudo-macro (see Table 5-4).

The CLI will therefore execute the true code path only if all three arguments are equal (condition #2).

Table 5-4. Conditional Arguments

Condition	Resolution	Code Path
1. %1%=%2% and %2%<%>3%	[!EQUAL,YES,NOT]	False
2. %1%=%2% and %2%=%3%	[!EQUAL,YES,YES]	True
3. %1%<%>2% and %2%=%3%	[!EQUAL,NO,YES]	False
4. %1%<%>2% and %2%<%>3%	[!EQUAL,NO,NOT]	False

Nested Conditionals

You may nest conditional pseudo-macros many levels deep. The CLI uses the following algorithm for determining which [!EQUAL]s and [!NEQUAL]s match up with which [!END]s.

1. First it looks for all [!EQUAL]/[!END] and [!NEQUAL]/[!END] pairs that do not have a conditional pseudo-macro between them, (the innermost pair).
2. The CLI then removes from consideration all conditional routines found in step 1, and starts over.
3. The CLI will repeat steps 1 and 2 until it accounts for all conditional pseudo-macros. If there are too many [!END]s, the CLI will execute part of the macro and return an error message. If there are not enough [!END]s, the CLI will return a special prompt when you execute the macro (see the section on unbalanced conditionals in this chapter).
4. If there are any [!ELSE]s that do not fall within a conditional routine, the CLI will issue an error message.

Figure 5-2 illustrates how the CLI evaluates nested pseudomacros.

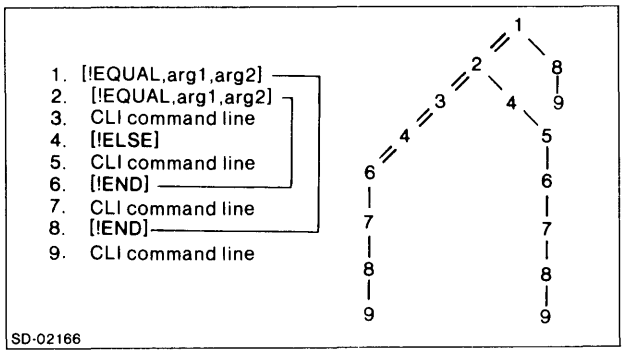


Figure 5-2. Nested Conditionals

Unbalanced Conditionals

If you attempt to execute a macro that does not have enough [!END]s in it, the CLI will issue one of two prompts — either \) or !). In both cases you should type [!END] to close the loop. The \) prompt signifies that the CLI is in a false code path, whereas the !) prompt signifies a true code path. This means that you can type commands next to the !) prompt. The CLI will execute them when you close the loop with a [!END]. If you still get the \) or !) prompt after you have entered a [!END], it means that you have more than one open loop. You should keep entering [!END]s until the CLI finishes executing the macro and returns a normal prompt.

Pseudo-Macro Syntax

The syntax for pseudo-macros is the same as for macros except when using parentheses.

Parentheses in Conditional Pseudo-Macros

Parentheses within a conditional pseudo-macro indicate grouping. Use parentheses to group together an expression that has separators within it. This is particularly useful for evaluating STRING, which may contain several words separated by separators. Suppose, for instance, that you want to write a macro that tests whether STRING equals INPUT FILE ERROR and then performs a certain execution if it does. If you write:

```
[!EQUAL,[!STRING],INPUT FILE ERROR]
```

the CLI will return an error telling you that there are too many arguments. This is because the [!EQUAL] pseudo-macro takes only two arguments and the CLI treats each word in the expression INPUT FILE ERROR as a separate argument. If, however, you surround the expression with parentheses, the CLI will evaluate it as a single argument. The following command will execute properly.

```
[!EQUAL,([!STRING]),(INPUT FILE ERROR)]
```

Note that you must surround [!STRING] with parentheses as well because both arguments must either be enclosed in parentheses or not. If only one is enclosed, the expression will always be false.

The CALCULATOR Macro

This macro (see Figure 5-3) performs the following arithmetic operations:

- Addition of two integers
- Subtraction of two integers
- Multiplication of two integers
- Division of two integers
- Conversion of a decimal number to an octal number
- Conversion of an octal number to a decimal number

You specify what operation you want by entering the macro name with the appropriate switch and argument(s). Table 5-5 shows the syntax for the CALCULATOR macro.

Examples

To add 3 and 51 with the CALCULATOR macro, type

)CACULATOR/A 3 51↵

The CLI returns with

Adding 51 to 3. Sum is 54
)

To subtract 10 from 20, type

) CALCULATOR/S 20 10↵

The CLI returns with

Subtracting 10 from 20. Result is 10
)

To multiply 10 by 35, type

) CALCULATOR/M 10 35↵

The CLI returns with

Multiplying 10 by 35. Product is 350
)

To divide 29 by 3, type

) CALCULATOR/D 29 3↵

The CLI returns with

Dividing 29 by 3. Quotient is 9 — and remainder is 2
)

To convert 16 base 10 to base 8, type

)CALCULATOR/O 16↵

The CLI returns with

Octal value of 16 is 20
)

To convert 16 base 8 to base 10, type

)CALCULATOR/DEC 16↵

The CLI returns with

Decimal value of 16 is 14
)

Table 5-5. CALCULATOR Macro Syntax

Macro Call	Result
CALCULATOR/A X Y	Entering the CALCULATOR macro with the /A switch and two integers returns the result of the addition of the two integers. You must enter two integers with the /A switch.
CALCULATOR/S X Y	Entering the CALCULATOR macro with the /S switch and two integers returns the result of the subtraction of the second argument from the first. You must enter two integers with the /S switch.
CALCULATOR/M X Y	Entering the CALCULATOR macro with the /M switch and two integers returns the result of the multiplication of the two integers. You must enter two integers with the /M switch.
CALCULATOR/D X Y	Entering the CALCULATOR macro with the /D switch and two integers returns the result of the division of the first argument by the second argument. You must enter two integers with the /D switch.
CALCULATOR/O X	Entering the CALCULATOR macro with the /O switch and one base 10 integer returns the equivalent integer in base 8
CALCULATOR/DEC X	Entering the CALCULATOR macro with the /DEC switch and one base 8 integer returns the equivalent integer in base 10.

```
[!EQUAL,(%1%),()]
COMMENT This pseudo-macro tests to see if you entered any arguments along
COMMENT with the macro name.  If you invoke this macro without any arguments,
COMMENT the following lines appear on your screen describing what the
COMMENT macro can do:
WRITE This macro -- %0% -- does arithmetic operations using integer
arguments.
WRITE To add use the /A switch and two integers  -- e.g., %0%, 45, 89
WRITE To subtract use the /S switch and two integers -- e.g., %0%/S, 39 22
WRITE To multiply use the /M switch and two integers -- e.g., %0%/M, 45 37
WRITE To divide use the /D switch and two integers -- e.g., %0%/D, 400 26
WRITE To convert decimal to octal use /O and one integer -- e.g., %0%/O, 29
WRITE To convert octal to decimal use /DEC and one integer -- e.g. %0%/DEC 44
[!ELSE]
COMMENT If you entered any arguments along with the macro name, the
COMMENT following pseudo-macros test to see if you entered /A for
COMMENT addition, /S for subtraction, /M for multiplication or /D for
COMMENT division as macro switches.
        [!EQ,%0/%,/A]
        WRITE
        WRITE Adding %2% to %1%. Sum is [!UADD,%1%,%2%]
        WRITE
        [!ELSE]
          [!EQ,%0/%,/S]
          WRITE
        WRITE Subtracting %2% from %1%. Result is  [!USUB,%1%,%2%]
        WRITE
          [!ELSE]
            [!EQ,%0/%,/M]
        WRITE
        WRITE Multiplying %1% by %2%. Product is  [!UMUL,%1%,%2%]
        WRITE
            [!ELSE]
              [!EQ,%0/%,/D]
        WRITE
        WRITE Dividing %1% by %2%. Quotient is [!UDIV,%1%,%2%] -- and
        WRITE remainder is [!UMODULO,%1%,%2%]
        WRITE
              [!ELSE]
COMMENT And these pseudo-macros test to see if you entered
COMMENT either /O for octal or /DEC for decimal as a macro switch.
                [!EQ,%0/%,/O]
        WRITE
        WRITE Octal value of %1% is [!OCTAL,%1%]
        WRITE
                [!ELSE]
                  [!EQ,%0/%,/DEC]
        WRITE
        WRITE Decimal value of %1% is [!DECIMAL,%1%]
        WRITE
                  [!ELSE]
COMMENT The following seven !END pseudo-macros, one for the beginning
COMMENT !EQUAL pseudo-macro and one for each of the six !ELSE
COMMENT pseudo-macros, tell the CLI to close each pseudo-macro.
                  [!END]
                [!END]
              [!END]
            [!END]
          [!END]
        [!END]
[!END]
```

Figure 5-3. The CALCULATOR Macro

End of Chapter

Chapter 6
CLI Commands, Pseudo-Macros, Languages and Systems Utilities

This chapter presents descriptions of CLI commands, pseudo-macros, languages, and system utilities, in alphabetical order.

Before you proceed, you may want to review the notation conventions described in the preface; these will help you understand the command format. In the examples, all user input lines begin with the CLI prompt) and end with a NEW LINE ↓; all lines without a CLI prompt are system responses.

Pseudo-Macros

Some CLI pseudo-macros, such as [!SEARCHLIST] and [!DATE], return environmental settings or system variables. Others, like [!EQUAL], allow you to specify conditional execution of commands. Still others, like [!OCTAL], convert values.

NOTE: Throughout this book, but specifically within this chapter, we use brackets to enclose both optional entries and pseudo-macros, for example, *[pathname]* and [!TIME]. Do not enter the brackets around an optional entry, they just set off the entry. The brackets around a pseudo-macro are part of the command, and you must enter them.

System Utilities and Languages

This chapter also reviews the most frequently used system utilities and languages. It contains brief descriptions of the CLI command lines used to invoke these utilities and languages. Many of these have their own manuals that contain more complete descriptions. To invoke a utility or language, begin the command line with the XEQ (or EXECUTE) command; this creates a process to execute the utility or language and blocks the calling process. (Note that some of the utilities and languages have macros that you use to invoke them. When you use a macro, you do not begin the command line with XEQ.) The CLI resumes execution when the utility or language has finished. To interrupt an executing utility, see *Control Characters* in Chapter 1.

Table 6-1 lists each CLI command, pseudo-macro, language, and utility command alphabetically, within the following eight sections:

- File Maintenance
- Process Control
- CLI Environment
- System Management
- Batch and Spooler Queue
- Miscellaneous
- Pseudo-Macros
- Utilities, and Languages

In all but the last section, the table shows sample formats for the commands. For details on any command, see its description later in the chapter.

Table 6-1. Command, Pseudo-Macro, Utility, and Language Summary by Category

File Maintenance

Command and Sample Format	Meaning
ACL pathname */user,access/[user,access/...*	Set or display a file's Access Control List.
COPY dest-file sourcefile */sourcefile/...*	Copy a file to dest-file.
CREATE pathname */resolution-pathname/*	Create a file.
DELETE pathname */pathname/...*	Delete a file.
DISMOUNT linkname */message/*	Ask operator to dismount a tape.
DUMP dumpfile */source-pathname/...*	Dump one or more directories or files.
FILESTATUS */pathname/...*	Display the status of one or more files.
LOAD dumpfile */source-pathname/...*	Load one or more dumped files.
MMOVE diskette-name */filename.../*	Transfers one or more files.
MOUNT linkname message	Mount a tape.
MOVE dest-dir */sourcefile/...*	Move copies of one or more files.
OPERATOR [*ON* | *OFF*]	Turn on or off the CLI's ability to load from, label, and dump to labeled diskettes.
PATHNAME pathname	Display a complete pathname starting at the root directory.
PERMANENCE pathname [*ON* | *OFF*]	Set or display a file's permanence attribute.
RENAME pathname newname	Rename a file.
REVISION pathname */field1/.field2/.field3/.field4////*	Set or display a program's revision number.
REWIND {tape-unit | linkname} [*tape-unit* | *linkname*]...	Rewind one or more tapes.
SPACE [*control-point-directory /new-max-size/* | *logical-disk*]	Set or display space unused and used in directory.
TYPE pathname */pathname/...*	Type the contents of a file.

(continues)

LOGFILE

LOGFILE contains the pathname of the current log file. If there is a file, it contains a copy of all input to @INPUT and all output to @OUTPUT. (Output from the TYPE command, however, does not get logged). Note that this applies only to the CLI. It does not apply to input or output from any programs executed from the CLI. You can set and display the log file with the LOGFILE command.

Working Directory

The working directory is your placemark in the directory tree. Chapter 2 contains an explanation of the working directory. (If you need a more comprehensive description of the working directory, see *Learning to Use Your Advanced Operating System* or *Learning to Use Your AOS/VS System.*)

You set and display the working directory with the CLI command DIRECTORY. You insert the working directory's pathname in a command line with the !DIRECTORY pseudo-macro (see Chapter 6).

Search List

The search list is a list of directory files that the system searches when it can't find a file in the working directory. Chapter 2 describes the search list. (For a broader discussion of this mechanism, see *Learning to Use Your Advanced Operating System* or *Learning to Use Your AOS/VS System.*)

You set and display the current search list with the CLI command SEARCHLIST. You can insert the search list in a command line with the !SEARCHLIST pseudo-macro (see Chapter 6).

Default ACL

Initially, the system sets a user's default ACL to username, OWARE. Using the DEFACL command, however, you can set your default ACL to whatever you want. This means that whenever you create a file, it will automatically take your default ACL.

You set and display the current default ACL with the CLI command DEFACL. You can insert the default ACL in a command line with the !DEFACL pseudo-macro (see Chapter 6).

See Chapter 2 for more information about ACLs and Chapter 6 for information about the DEFACL command and the !DEFACL pseudo-macro.

STRING

The CLI maintains a 127-character buffer called STRING. One use for this buffer is that you can divert a program's termination message, normally displayed on the terminal, to STRING by using the /S switch on the command you use to invoke the program. This feature is extremely useful in macros and batch jobs. Another use for this buffer is in holding a text string within a macro.

You set and display the current STRING with the CLI command STRING. You can insert the contents of STRING in a command line with the !STRING pseudo-macro (see Chapter 6).

PROMPT

When the CLI is ready for your input, it displays the prompt) on @OUTPUT after it executes the CLI commands contained in the prompt buffer. The prompt buffer can hold a maximum of eight CLI commands. By default, the prompt is null; the CLI simply displays) when it is ready for input.

The PROMPT command can be a handy tool. For instance, if you move through your directory tree structure from file to file, it is easy to forget what your currrent working directory is. But having the PROMPT command execute the DIRECTORY command everytime you hit NEW LINE, the screen always displays the pathname of your current working directory.

You set and display the current prompt setting with the CLI command PROMPT (see Chapter 6).

CHARACTERISTICS

A console's characteristics control the way it interprets input and sends output. The environment preserves the characteristics of your console.

You can set and display a device's current characteristics with the CLI command CHARACTERISTICS (see Chapter 6).

Exceptional Conditions

When you enter an invalid command line or if you try to execute a syntactically correct line in an improper environment, you create an exceptional condition. The action that the CLI takes depends on the class of the exceptional condition and the current setting of that class.

CLASS1

CLASS1 exceptional conditions occur for any command that would change any of the parameters that make up the CLI environment if the command succeeded. If it doesn't succeed, the CLI returns a CLASS1 exceptional condition. For example, if you issue the POP command while in level 0, you will evoke a CLASS1 exceptional condition. Likewise, if you attempt to change your working directory to a nonexistent directory or to a directory to which you do not have Execute access, then you will cause a CLASS1 exceptional condition.

The action that the CLI takes on CLASS1 exceptional conditions depends on the current CLASS1 setting. You can set CLASS1 to any of the following: IGNORE, WARNING, ERROR, or ABORT. Table 4-1 summarizes CLI action on CLASS1 exceptional conditions. By default, CLASS1 is set to ERROR in interactive mode and to ABORT in batch jobs.

You set and display the current CLASS1 setting with the CLASS1 command. You override the current setting for the duration of any CLI command with the /1= command switch.

Table 4-1. Exceptional Condition Settings

Setting	Effect
IGNORE	No effect. The CLI ignores the exceptional condition and continues processing as best it can.
WARNING	The CLI displays a warning message and continues processing. The only difference between IGNORE and WARNING is the message.
ERROR	Execution ceases for the current command and the CLI displays an error message. In macros and multiple-command input lines, the CLI clears the command buffer and prompts for a new command line.
ABORT	ABORT terminates the CLI and control returns to the CLI's father. If the CLI's father is EXEC, it logs you off the system.

CLASS2

CLASS2 exceptional conditions occur for any command that does not alter the environment. When such a command fails, the CLI responds with whatever action the CLASS2 parameter is set to. For example, if you attempt to rename a nonexistent file, then you will cause the CLASS2 exceptional condition. Likewise, if you issue a DELETE command for a file to which you do not have Write access, you will evoke a CLASS2 exceptional condition.

The action that the CLI takes on CLASS2 exceptional conditions depends on its current setting. You can set CLASS2 to any of the following: IGNORE, WARNING, ERROR, or ABORT. Table 4-1 summarizes CLI action on CLASS2 exceptional conditions. By default, CLASS2 is set to WARNING in both interactive and batch mode.

You set and display the current CLASS2 setting with the CLASS2 command. You override the current CLASS2 setting for the duration of any CLI command with the /2= command switch.

Two Examples Using CLI Environment Levels

The following two examples suggest two ways to use CLI environment levels.

Example One

The PUSH and POP commands are especially useful in macros, since they allow you to alter your environment when you begin macro execution and to restore the original environment when you return to interactive mode. The macro file SAMPLE, contains the following command lines.

```
1     PUSH
2     SQUEEZE OFF
3     CLASS1 ABORT
4     CLASS2 WARNING
5     LISTFILE %1%
6     DATAFILE %2%
7     DIRECTORY :UDD:USER:MDIR
8     SEARCHLIST :UDD:USER:NDIR [!SEARCHLIST]
9     XEQ/S MYPROG1
10    !EQUAL,[!STRING],2]
11    XEQ MYPROG2
12    [!ELSE]
13    XEQ MYPROG3
14    [!END]
15    POP
```

Figure 4-1. The SAMPLE Macro

Table 6-1. Command, Pseudo-Macro, Utility, and Language Summary by Category

Process Control

Command and Sample Format	Meaning
ASSIGN character-device *[character-device]...*	Assign a character-device (nonspooled) for your exclusive use.
BLOCK { username:procname / process-ID } [*username:procname / process-ID*] ...	Block a process.
BREAKFILE { username:procname / process-ID }	(AOS/VS only) Set or display the current BREAKFILE format of a process ring.
BYE *[argument]...*	Terminate the CLI at your console.
CHAIN pathname *[argument-to-new-program]* ...	Overwrite your CLI with the program named in pathname.
CHECKTERMS	Check for the termination of a son process.
CONNECT { username:procname / process-ID }	Establish a customer-server connection.
DEASSIGN character-device *[character-device]...*	Deassign a previously assigned character-device.
DEBUG pathname *[argument-to-new-program]...*	Execute the program named in pathname with the Debugger.
DISCONNECT process-ID	Break a customer server connection.
EXECUTE pathname *[argument-to-new-program]...*	Execute pathname.
HOST *[hostID]* ...	Display a system's hostname.
PAUSE { seconds / seconds.milliseconds }	Delay the CLI.
PRIORITY [{ *username:procname / process-ID* } *[new-priority]*]	Set or display a process's priority.
PROCESS pathname *[argument-to-new-process]...*	Create a process.
PRTYPE [{ *username:procname / process-ID* } [*PREEMPTIBLE / RESIDENT / SWAPPABLE*]]	Set or display a process's type.
RUNTIME [*username:procname / process-ID*] ...	Display a process's runtime information.

(continued)

Table 6-1. Command, Pseudo-Macro, Utility, and Language Summary by Category

Process Control (continued)

Command and Sample Format	Meaning
TERMINATE {username:procname / process-ID} [*username:procname / process-ID*]...	Terminate a process.
TREE [*username:procname / process-ID*]...	Display a process's family tree.
UNBLOCK {username:procname / process-ID} [*username:procname / process-ID*]...	Unblock a process.
WHO [*username:procname / process-ID*] ...	Display process information.
XEQ pathname *[argument-to-new-program]...*	Execute a program.

CLI Environment

Command and Sample Format	Meaning
CHARACTERISTICS *[device]...*	Display or set device characteristics.
CLASS1 [*IGNORE / WARNING / ERROR / ABORT*]	Display or set CLASS1 severity level.
CLASS2 [*IGNORE / WARNING / ERROR / ABORT*]	Display or set CLASS2 severity level.
CURRENT	Display the current environment's setting.
DATAFILE *[pathname]*	Display or set the current DATAFILE.
DEFACL *[username, access]...*	Display or set the default access control list.
DIRECTORY *[pathname]*	Display or set the working directory.
LEVEL	Display the current environment level.
LISTFILE *[pathname]*	Display or set the current LISTFILE.
LOGFILE *[pathname]*	Display or set the current logfile.
PERFORMANCE	Display CLI statistics.
POP	Return to the next-higher environment level.
PREVIOUS	Display the previous environment's setting.
PROMPT *[command]...*	Display or set the CLI's PROMPT setting.

(continued)

Table 6-1. Command, Pseudo-Macro, Utility, and Language Summary by Category

CLI Environment

Command and Sample Format	Meaning
PUSH	Descend one environment level.
SCREENEDIT [ON / OFF]	Set or display the current SCREENEDIT mode.
SEARCHLIST *[pathname]...*	Display or set the search list.
SQUEEZE [ON / OFF]	Display or set the SQUEEZE mode.
SUPERPROCESS [ON / OFF]	Set or display the SUPERPROCESS setting.
SUPERUSER [ON / OFF]	Display or set the SUPERUSER mode.
STRING *[argument]...*	Display or set the current STRING.
TRACE	Display or set the current trace mode.
VAR0 *[argument]* through VAR9 *[argument]*	Display or set current value of one of the ten CLI variables.

System Management

Command and Sample Format	Meaning
(Operator Commands; also see Utilities).	
BIAS *[minimum-no[maximum-no]]*	Display or set the system's bias factor.
CONTROL ipcport argument *[argument]...*	Send a control message to a process.
CPUID	(AOS/VS only.) Display the CPU Identification.
DATE *[date]*	Display or set the current system date.
INITIALIZE physical-unitname *[physical-unitname]...*	Graft a logical disk into the working directory.
LOGEVENT message	Enter a message in the system log file.
RELEASE logical-disk *[logical-disk]*	Release a logical disk from the working directory.
SYSID *[argument]...*	Set or display the system identifier.
SYSINFO	Display current system information.
SYSLOG *[filename]*	Display the current state of the log or start or stop writing to the system usage log file.
TIME *[new-time]*	Display or set the current system time.

(continued)

Table 6-1. Command, Pseudo-Macro, Utility, and Language Summary by Category

Queue Facilities

Command and Sample Format	Meaning
ENQUEUE device-pathname *[pathname]...*	Queue an entry to a spooler queue.
QBATCH argument *[argument]...*	Create and submit a batch job file.
QCANCEL {seq-no / jobname} [*seq-no* / *jobname*]...	Delete an entry from a queue.
QDISPLAY *[hostname]*	Display queue information.
QFTA/DESTINATION=pathname source-pathname	Place an entry on the FTA queue.
QHOLD {seq-no / jobname} [*seq-no* / *jobname*]...	Hold a queue entry.
QPLOT pathname *[pathname]...*	Place an entry on the plotter queue.
QPRINT pathname *[pathname]...*	Place an entry on the line printer queue.
QPUNCH pathname *[pathname]...*	(AOS only.) Place an entry on the paper tape punch queue.
QSNA pathname *[pathname]...*	Place an entry on a Systems Network Architecture (SNA) Queue.
QSUBMIT pathname *[pathname]...*	Place an entry on a batch or spooler queue.
QUNHOLD {seq-no / jobname} [*seq-no* / *jobname*]...	Release a held entry.

Miscellaneous Commands

Command and Sample Format	Meaning
Comment	Include a comment.
HELP *[item]...*	Display information about one or more CLI *items*.
MESSAGE errorcode *[errorcode]...*	Display text message corresponding to errorcode.
PREFIX *[argument]...*	Display or set the prefix string.
SEND {process-ID / username:processname / consolename} message	Send a message to another terminal.
WRITE *[argument]...*	Display arguments.

(continued)

Table 6-1. Command, Pseudo-Macro, Utility, and Language Summary by Category

Pseudo-Macros

Pseudo-Macros	Meaning
[!ACL pathname]	Expand to a file's Access Control List.
[!ASCII octal-number */octal-number/*...]	Expand to characters corresponding to octal arguments.
[!CONSOLE]	Expand to the console name.
[!DATAFILE]	Expand to the current data file pathname.
[!DATE]	Expand to the current system date.
[!DECIMAL octal-number]	Convert a number from octal to decimal representation.
[!DEFACL]	Expand to the user default Access Control List.
[!DIRECTORY]	Expand to the current working directory pathname.
[!EDIRECTORY pathname */pathname/*...]	Expand to the directory portion of a pathname.
[!EEXTENSION pathname */pathname/*...]	Expand to the extension portion of a pathname.
[!EFILENAME pathname */pathname/*...]	Expand to the filename portion of a pathname.
[!ELSE]	Execute CLI commands conditionally.
[!ENAME pathname */pathname/*...]	Expand to the name portion of a pathname.
[!END]	End a conditional input sequence.
[!EPREFIX pathname */pathname/*...]	Expand to the prefix portion of a pathname.
[!EQUAL argument1,argument2]	Execute CLI commands conditionally.
[!EXPLODE argument */argument/*...]	Expands arguments into single-character, CLI-acceptable arguments.
[!FILENAMES */pathname/*...]	Expand to a list of filenames.
[!HID */hostname/*]	Expand to a host ID.
[!HOST */hid/*]	Expand to a hostname.
[!LEVEL]	Expand to the current CLI environment level number.
[!LISTFILE]	Expand to the current list file pathname.
[!LOGON]	Determine if user is logged on under EXEC and, if so, expand to CONSOLE or BATCH.

(continued)

Table 6-1. Command, Pseudo-Macro, Utility, and Language Summary by Category

Pseudo-Macros (continued)

Pseudo-Macros	Meaning
[!NEQUAL argument1,argument2]	Execute macro commands conditionally.
[!OCTAL decimal-number]	Convert a number from decimal to octal representation.
[!OPERATOR]	Expand to ON or OFF depending on whether operator is on or off duty.
[!PATHNAME pathname]	Expand to a file's full pathname.
[!PID]	Expand to your CLI's process ID.
[!READ argument *[argument]*...]	Display text on @OUTPUT and expand to arguments from @INPUT.
[!SEARCHLIST]	Expand to the search list.
[!SIZE pathname]	Expand to the size, in bytes of a file.
[!STRING]	Expand to the STRING setting.
[!SYSTEM]	Expand to the name of the operating system.
[!TIME]	Expand to the current system time.
[!UADD argument1,argument2]	Expand to argument1 plus argument2.
[!UDIVIDE argument1,argument2]	Expand to argument1 divided by argument2.
[!UEQ argument1,argument2]	Execute macro commands conditionally.
[!UGE argument1,argument2]	Execute macro commands conditionally.
[!UGT argument1,argument2]	Execute macro commands conditionally.
[!ULE argument1,argument2]	Execute macro commands conditionally.
[!ULT argument1,argument2]	Execute macro commands conditionally.
[!UMODULO argument1,argument2]	Expand to argument1 modulo argument2.
[!UMULTIPLY argument1,argument2]	Expand to argument1 multiplied by argument2.
[!UNE argument1,argument2]	Execute macro commands conditionally.
[!USERNAME]	Expand to the CLI's username.
[!USUBTRACT argument1,argument2]	Expand to argument1 minus argument2.
[!VAR0] through [!VAR9]	Expand to the current value of one of the CLI variables.

(continued)

Table 6-1. Command, Pseudo-Macro, Utility, and Language Summary by Category

System Utilities and Languages	
Utility or Language	**Function**
You must invoke some of these utilities with the CLI command XEQ (or EXECUTE), or PROCESS. Others use macros and do not require XEQ.	
XEQ AOSGEN	(AOS only) Generate a new AOS system.
XEQ APL	(AOS/VS only) Invoke the APL interpreter.
XEQ BASIC	Invoke the BASIC interpreter.
XEQ BIND	(AOS only) Bind object modules to form an executable program.
XEQ BRAN	(AOS/VS only) Analyze an AOS/VS break file.
CBIND	(AOS only) Bind COBOL object modules into an executable program.
XEQ CC	(AOS/VS only) Compile a C program module.
CLINK	(AOS/VS only) Link object modules into an executable COBOL program.
COBOL	Compile a COBOL source file.
XEQ CONVERT	Convert an RDOS.RB binary file to an AOS.OB binary file.
XEQ DEDIT	(AOS only) Edit disk file locations.
XEQ DGL	Compile a DG/L™ source file.
XEQ DISPLAY	Print a file in octal and ASCII.
XEQ DUMP_II	(AOS/VS only) Dump one or more directories or files.
XEQ FCU	Set horizontal tabs and vertical form settings.
XEQ FED	(AOS/VS only) Edit disk file locations.
XEQ FILCOM	Compare two files.
FORT4	(AOS only) Compile a FORTRAN IV source file.
F5	Compile a FORTRAN 5 source file.
F5LD	Link object modules into an executable FORTRAN 5 program.
F77	Compile a FORTRAN 77 source file.
F77LINK	Link object modules into an executable FORTRAN 77 program.

(continued)

Table 6-1. Command, Pseudo-Macro, Utility, and Language Summary by Category

System Utilities and Languages (continued)	
Utility or Language	**Function**
XEQ LABEL	Prepare a mag tape with a volume label.
XEQ LFE	Edit library file.
XEQ LINEDIT	(AOS only) Edit ASCII text.
XEQ LINK	Link object modules to form an executable program file.
XEQ LOAD_II	(AOS/VS only) Load one or more directories or files.
XEQ MASM	Assemble source files to produce an object file.
XEQ MASM16	Assemble 16-bit source files on an AOS/VS system.
MERGE	Invoke the Sort/Merge utility to merge two files.
XEQ MKABS	(AOS only) Convert an RDOS save file to an absolute binary file.
XEQ MPL	Invoke the macro processor for procedural languages.
XEQ PED	Display the process environment.
XEQ PL1	Compile a PL/I source file.
PL1LINK	Bind object modules into an executable PL/I program.
XEQ PREDITOR	Invoke the User Profile Editor.
XEQ RDOS	Read or write an RDOS dumpfile or disk.
XEQ REPORT	Print the system log file.
RIC	Invoke the RPGII Interpretive Compiler.
ROC	Invoke the RPGII Optimizing Compiler.
RPG	Compile an RPG II source file.
XEQ SCOM	Compare two ASCII text files.
XEQ SED	Edit an ASCII text file.
XEQ SLB	(AOS only) Build a shared library.
SORT	Invoke the Sort/Merge utility to sort a file.
XEQ SPEED	Edit an ASCII text file.
XEQ SWAT	Invoke the high-level interpretive debugger.
XEQ VSGEN	(AOS/VS only) Generate a new AOS/VS system.

(concluded)

ACL *Command*

Set or display the access control list for a file.

Format

ACL pathname *[user access]...*

To display a file's ACL, supply pathname as the only argument to the ACL command. To set or change a file's ACL, specify user and access as well as pathname. You may use templates in the pathname argument and in the user argument. Separate the arguments with a valid CLI separator: one or more blanks, one or more tabs, one comma, or any combination of these (e.g., one comma and one or more blanks).

The CLI displays the access control list (ACL) in the following format:

username-template access-types username-template access-types ...

where *access-types* is a string of one or more of the following characters:

Character	Meaning
O	Owner access
W	Write access
A	Append access
R	Read access
E	Execute access

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/D		Give the file the user's default ACL (takes pathname argument only).
/K		Delete ACL; this denies everyone except Superusers access to the file until the ACL changes again (takes pathname argument only).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/V		Display the filename with the ACL.

Argument Switches

None

Example

```
) ACL TEST.PR ↵
JONES,R PROJ.-,RE
) ACL TEST.PR,JONES,WARE,PROJ.-,RE ↵
) ACL/V TEST.PR ↵
TEST.PR  JONES,WARE PROJ.-,RE
)
```

The first ACL command displays the access control list for file TEST.PR. The second ACL command sets a new access control list for that file, and the third command displays the new access control list preceded by the filename.

[!ACL] *Pseudo-Macro*

Expand to a file's access control list.

Format

[!ACL pathname]

This pseudo-macro requires one pathname argument.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE FILE'S ACL IS [!ACL FILE] )
FILE'S ACL IS COLLATE_DEBTS,OWARE
)
```

The CLI first evaluates the pseudo-macro [!ACL FILE], then writes the resulting argument list on the terminal.

AOSGEN *Utility*

Generate a new operating system (AOS only).

AOSGEN is the Advanced Operating System generation program. We describe AOSGEN further in *How to Load and Generate Your Advanced Operating System* (093-000217).

APL *Language*

Invoke the APL interpreter (AOS/VS only).

Format

XEQ APL *[initial-workspace-pathname]*

or

APL *[initial-workspace-pathname]*

You can use either the XEQ APL command line or the APL.CLI macro provided to you to interpret an APL program. If you chose to use the XEQ APL command you must have the :APL directory in your searchlist.

You may specify an initial workspace. If you do not, the interpreter looks by default for a workspace named CONTINUE. The interpreter loads the workspace if it exists. If it does not exist, execution begins in a clear workspace.

Some of the APL switches require a value that identifies your terminal or input device to the system. The following codes are acceptable for the switches that take a terminal type (term-type) value:

Term-type	Meaning
0	Batch (default for disk files)
1	605x or D200 compatible terminal without the APL character set
2	6110 APL display terminal (default for CRTs)
3	TP2 model 6193 with the APL character set downline loaded (default for hardcopy devices)
4	APL/ASCII typewriter pairing terminal
5	APL/ASCII bit pairing terminal
6	D450 or D460 compatible terminal with the APL character set downline loaded

For terminal types 3 and 6, the terminal should have been downline loaded with the APL character set before invocation of APL. A good time to do this is when the terminal is powered on. To downline load the terminal, simply issue the CLI command

) COPY/B @CONSOLE :APL:filename)

where filename is APL_LPT_CLEANUP for type 3 or APL_D450_CLEAN_UP for type 6. The :APL directory contains both of these files.

For terminal types 4 and 5, APL sends the ASCII shift-out character (ASCII SO) to your terminal. This changes your character set to the APL character set. When you exit APL, APL sends the ASCII shift-in character (ASCII SI), which changes the character set back to ASCII. If your terminal does not respond to the shift characters, or if you use a keyboard switch to change character sets, you should also use the /INS, /ONS, or /LNS switches to suppress the shift characters.

See the *APL Reference Manual (AOS/VS)* (093-000274) for more information.

APL (continued)

APL Switches

/ESC	Do not interpret the ESCAPE key as ↑C↑A.
/I=pathname	Specify pathname as the input file. If you do not use this switch, the default input file is @INPUT.
/INS	Do not write ASCII shift characters to the input file.
/ITT=term-type	Specify the input file's terminal type. See the previous list for the possible values for term-type.
/L=pathname	Specify pathname as the log file. If you do not use this switch, the default is no log file.
/LNS	Do not write ASCII shift characters to the log file.
/LTT=term-type	Specify the log file's terminal type. See the previous list for the possible values for term-type.
/MINUS	Print APL's overbar as – on all output.
/O=pathname	Specify pathname as the output file. If you do not use this switch, the default output file is @OUTPUT.
/ONS	Do not write ASCII shift characters to the output file.
/OTT=term-type	Specify the output terminal's type. See the previous list for the possible values for term-type.
/PW=integer	Set PW (the page width) to integer number of characters when a clear workspace is activated.
/SLX	Suppress execution of LX (the latent expression).
/TAKE	Use the uparrow (↑) as an error indicator.
/WSLIMIT=integer	Specify the maximum amount of space, in bytes, that you wish your workspace to use.

Example

```
) APL /ITT=6/OTT=6/L=LOGGING.FILE  PROG2 ⍙
```

Invoke the APL interpreter, loading PROG2.WS as the initial workspace. In addition, use LOGGING.FILE as the log file. The terminal output (/OTT) and input (/ITT) is a D460 (type 6).

[!ASCII] *Pseudo-Macro*

Expand to characters corresponding to octal arguments.

Format

[!ASCII octal-number *[octal-number]...*]

Each octal number must be a positive integer in the range 1 to 377.

Macroname Switches

None

Argument Switches

None

Example

You can use !ASCII to enter special characters that the CLI wouldn't normally interpret correctly. For instance, if you want to use the WRITE command to ring your console's bell, you cannot type the bell character (CTRL-G) into a WRITE command. If you tried to do this, the CLI would merely echo a ^G (CTRL-G).

```
) WRITE [!ASCII 207] ⍙
)
```

This example shows you how to use !ASCII to include a bell character with the parity bit set.

ASSIGN *Command*

Assign a character device for your exclusive use.

Format

ASSIGN character-device *[character-device]...*

character-devices include the card reader, consoles, etc.

After you ASSIGN a device, you control it until you either DEASSIGN it or log off the system. Note that you cannot ASSIGN a spooled device under the EXEC.

You may use templates in the character-device argument.

Command Switches

/1=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) ASSIGN @MTB )
)
.
.(you have exclusive use of the tape drive)
.
) DEASSIGN @CRA )
)
```

First gain control of the tape drive using the ASSIGN command, then release control of the tape drive using the DEASSIGN command.

BASIC *Utility*

Invoke the BASIC interpreter.

Format

For AOS:

XEQ BASIC

For AOS/VS:

XEQ BASIC *[program-pathname]*

BASIC is a programming language interpreter. You use the BASIC utility to create and execute BASIC programs. Several versions of BASIC are available. Different operating systems use different versions of the BASIC interpreter.

Under AOS/VS, you may provide a program pathname as an argument. The program must be a program file (type PRV), a BASIC core-image file (type BCI), or a BASIC source file. BASIC can use this pathname as an argument to its CHAIN command, which stops execution of the current program and loads and runs the specified program.

For more information on BASIC and the BASIC utility, see the documentation for the version of BASIC that runs under your operating system:

basic BASIC (069-000003) (AOS only)

Extended BASIC User's Manual (093-000065) (AOS only)

BASIC (AOS/VS) User's Manual (093-000252) (AOS/VS only)

BASIC Switches

/NOSIGN	(AOS/VS only) Do not output sign-on and sign-off messages.

Argument Switches

None

Example

```
) XEQ BASIC )
*
.
.(Enter BASIC commands)
.
* BYE )
)
```

Invoke the BASIC interpreter, enter BASIC commands, then exit from the BASIC interpreter.

BIAS *Command*

Set or display the system's bias factor.

Format

BIAS *[minimum number [maximum number]]*

Any process can display the system's bias factor but only PID 2 can set it. The default minimum is 0 and the default maximum is no limit. If you set only the minimum number, the maximum will automatically be set to no limit. See *How to Generate and Run AOS* (093-000217) or *How to Generate and Run AOS/VS* (093-000243) for a description of the system's bias factor.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) BIAS ↵
MINIMUM: 0, MAXIMUM: NONE
)
```

Displays the system's bias factor.

```
) BIAS 0 ↵
)
```

Sets the system's bias factor to MINIMUM: 0, MAXIMUM: NONE.

```
) BIAS 0 21 ↵
)
```

Sets the system's bias factor to MINIMUM: 0, MAXIMUM: 21.

BIND *Utility*

Bind object modules to form an executable program file (AOS only).

Format

XEQ BIND objectmodule *[argument]...*

You use the Binder to build an executable program file from object files.

The BIND utility names the first bound object file objectmodule. And unless you specify otherwise with the /P switch, the BIND utility names the program file objectmodule.PR.

[argument]... can be any of the following:

- Another object module
- A shared or unshared library name
- A symbol name or integer, with the appropriate switch
- A command file, with the /C switch

The command file specifies objects you want to bind as overlays, along with their switches. BIND places these files in overlay file objectmodule.OL, to correspond with program file objectmodule.PR. To bind overlays, you must build (CREATE) the command file from the binaries that you want to be overlays. See the *AOS Binder User's Manual* (093-000190) for more information.

BIND Switches

/B	Produce a symbol file listing, ordered alphabetically and numerically.
/E	Output the load map to @OUTPUT, even if you specified a listing file using the /L=pathname switch.
/H	Print all numbers in hexadecimal.
/I	Build a nonexecutable program file without a user status table, task control blocks, or other system tables. Do not scan the user runtime library (URT.LB). This switch can help check for BIND errors, such as multiply defined or undefined .ENT symbols.
/K=integer	Set the number of tasks to integer. This number overrides any .TSK pseudo-op statement included in the source file.
/L	Produce a listing file, using the current list file.
/L=pathname	Produce a listing file, using the file specified by pathname.
/M=integer	Save integer number of 1K-word pages of memory for shared library use.
/N	Do not scan the user runtime library (URT.LB).
/O	Allow load overwrites to occur.
/P=pathname	Name the program file pathname.PR. The default is objectmodule.PR.
/S	Produce a shared routine. You include this switch if you want to build a shared library.
/T=integer	Set decimal integer as the top of the shared area. BIND rounds this area to an even 1K boundary.
/Z=integer	Set decimal integer as the stack size for the program. By default, BIND allocates 30 words (decimal).

Argument Switches

/AM=integer	Set total overlay area to integer number of basic areas. This switch applies only to a right bracket in an overlay specification.
/B	Bind this shared library into the shared area. This switch applies only to a shared library.
/C	Specify that this file contains the objects to bind as overlays.
/D	Bind nonshared code in this module into the nonshared data area.
/H	Bind nonshared code in this module into the shared code area. If you append this switch to the name of a nonshared library, the utility will bind the records extracted from the library into the shared area.
/R	Issue a warning if any code in this module is not position independent.
/U	Write local symbols from this module to the symbol file. If the macroassembler produced this .OB file, you cannot use this switch unless you also specified /U to the macroassembler.

BIND (continued)

symbol/V=integer	Assign value integer to this accumulating symbol previously defined by pseudo-op .ASYM.
/X	(Used in conjunction with the /B switch.) Exclude this shared library routine that is in the library that the /B switch includes.
integer/Z	Set the ZREL base to this octal integer. If the current ZREL base exceeds this value, the system ignores the switch.

Example

```
) XEQ BIND/L=LFILE MYPROG MYLIB 100/Z ↵
```

This command line binds two objects, MYPROG and MYLIB, into program file MYPROG.PR. Page zero (ZREL) code will start at location 100_8. The listing goes to disk file LFILE.

```
) CREATE/I COMMANDFILE ↵
)) [OVLY1,OVLY2 OVLY3,OVLY4] ↵
))) ↵
)
```

This creates a command file containing the names of four object files. You can use this command file to create three overlays. Since no comma separates OVLY2 from OVLY3, the system will bind them into one overlay. To tell the system to reserve one overlay area in memory for the three overlays, enter the overlays in one set of brackets. Each will occupy this area as the program calls it. Each overlay should specify the same kind of relocation: shared or unshared.

```
) XEQ BIND/L OBJECT1 OBJECT2 COMMANDFILE&↵
&)/C ↵
```

This command line binds two objects into program file OBJECT1.PR, and the overlays in COMMANDFILE into overlay file OBJECT1.OL. The Binder listing goes into the current list file.

BLOCK *Command*

Block a process.

Format

BLOCK { username:procname | process-ID }

[*username:procname* | *process-ID*] ...

You must supply the process ID or the process name. The process you want to block must be an inferior process (unless you have the Superprocess privilege in which case you can block any process). procname must be a full process name. (See the appropriate programmer's manual (AOS or AOS/VS) for a description of the process name).

Command Switches

/1=	IGNORE | WARNING | ERROR | ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE | WARNING | ERROR | ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) BLOCK 19 ↵
)
```

Blocks the process with PID (process ID) 19.

```
) BLOCK SMITH:PROG1 ↵
)
```

Blocks the process named PROG1.

BRAN *Utility*

Analyze an AOS/VS break file (AOS/VS only).

Format

XEQ BRAN break-file-pathname
[symbol-table-pathname]

The BRAN utility produces a report from an AOS/VS break file, giving global information about the process and information for each active task. The process information includes the program type, memory usage, the current task (active when the process terminated), and the number of free tasks and active tasks. For each active task, the BRAN report describes the task identifier (task ID), the task's priority, and the values of the program counter (pc), the accumulators, and the stack pointers. In addition, the report lists the system call being serviced at the time of the termination.

If you cite the symbol table pathname as an argument, BRAN prints certain values (such as the program counter) symbolically.

Command Switches

/L=pathname — Write the break file report to pathname instead of @OUTPUT.

Example

```
) XEQ BRAN/L=REPORT & )
&) ?010.023_026_016.BRK MYPROG.ST )
)
```

Analyze the break file ?010.023_026_016.BRK from program MYPROG.PR, and write the break file report to file REPORT. Since the command line cites the symbol table MYPROG.ST, list the program counter symbolically.

BREAKFILE *Command*

Set or display the current BREAKFILE format of a process ring (AOS/VS only).

Format

BREAKFILE { username:procname / process-ID }

The BREAKFILE command enables a breakfile to be taken for the specified ring if the target process traps, executes ↑C↑E, or is the target of a TERM.

The BREAKFILE command can be issued against the calling process for all rings greater than or equal to the ring of the task making the call. No other privileges are required. This command can also be issued against another process if the calling process has Superprocess on or is a server and has a valid connection to the target process. In the case of a server issuing BREAKFILE against a customer, the specified ring to dump must be greater than or equal to the ring of the connection.

NOTE: The hierarchy structure is not sufficient to issue the BREAKFILE command. (i.e., a process cannot issue a BREAKFILE against its son unless it has Superprocess on, or has a valid connection to its son.)

Command Switches

/1= { IGNORE / WARNING / ERROR / ABORT } — Set CLASS1 to the specified severity level for this command.

/2= { IGNORE / WARNING / ERROR / ABORT } — Set CLASS2 to the specified severity level for this command.

/ALL — Specify that the breakfile created should contain the entire process space of the selected ring.

/DIRECTORY=pathname — Define the directory in which to create a breakfile. The breakfile created will have the default form ?PID.TIME.RING.BRK.

/FILENAME=pathname — Define the pathname to be used when creating a breakfile.

BREAKFILE (continued)

/KILL	Specify that the target process will not create a breakfile of the selected ring.
/L	Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname	Write CLI output to the file specified by **pathname** instead of to @OUTPUT.
/PREAMBLE	Specify that the breakfile of the target contain only a dump of the preamble area of the selected ring.
/Q	Set SQUEEZE to ON for this command.
/RING=N	Ring of target process's breakfile (default ring = 7).
/SHARED	Specify that the breakfile of the target process contain only a dump of the shared area of the selected ring.
/UNSHARED	Specify that the breakfile of the target process contain only a dump of the unshared area of the selected ring.

/KILL, /ALL, /SHARED, /UNSHARED are mutually exclusive.

/DIRECTORY= and /FILENAME= are mutually exclusive.

This command displays the current BREAKFILE format for the target process ring, when none of the following switches are selected: /KILL, /ALL, /PREAMBLE, /SHARED, /UNSHARED, /DIRECTORY /FILENAME.

Example

```
) BREAKFILE/ALL/FILENAME=:UDD:JACK: &
&) MYPROG.BRK MYPROG
```

This command tells the system that if process MYPROG traps, it is to take a memory image of all the ring 7 address space and to unite the breakfile to file :UDD:JACK:MYPROG.BRK in the working directory.

BYE *Command*

Terminate this CLI process.

Format

BYE *[argument]...*

The CLI returns any *argument* as a string to the father process.

Furthermore, if you issue the BYE command when you have sons, the CLI outputs the message

YOU HAVE SONS. DO YOU WANT TO TERMINATE?

and waits for a YES answer before terminating.

If you respond NO, the CLI does not terminate.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by **pathname** instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/WARNING /ERROR /ABORT		If you include any of these switches, the CLI will terminate, signalling the specified severity level. If you include any arguments to the command, the arguments will return to the calling process as a string.

Argument Switches

None

Example

) BYE ↓
AOS CLI TERMINATING 26-JUL-84 13:00:19

Terminates the CLI. BYE logs you off of the system if the CLI's father was EXEC. (Note that, under AOS/VS, the termination message reads *AOS/VS CLI TERMINATING* .)

CBIND *Language*

Bind object modules to form an executable COBOL program (AOS only).

Format

CBIND objectmodule *[argument]...*

The CBIND program invokes the AOS Binder utility to make COBOL object modules into an executable program.

objectmodule specifies the main program. If you do not provide a filename extension, CBIND assumes that the complete filename is objectmodule.OB. You may also include other arguments on the command line. An argument might specify a subprogram, shared library, accumulating symbol, or some other part of the loaded program. ICALL is the COBOL interface to the INFOS system (supplied by INFOS), which you need if you use INFOS indexed files. Either use LFE to add ICALL to URT.LB, or include ICALL on the CBIND command line.

For a complete description of the COBOL programming language and the CLI CBIND command line, see the *COBOL Reference Manual (AOS)* (093-000223).

CBIND Switches

/B	List the symbol table in alphabetical and numeric order.
/D	Bind in the COBOL debugger program. Load the COBOL program modules as unshared code.
/E	Output the load map to @OUTPUT, even if you specified another listing file using the /L=pathname switch.
/H	List all numbers in hexadecimal.
/I	Build a nonexecutable program file, lacking a UST, TCBs, and all other system databases.

CBIND (continued)

/K=integer	Allocate integer number of TCBs for multitask use, regardless of the number specified in a .TSK statement.
/L	Write a listing to the current list file.
/L=pathname	Write a listing to the file specified by pathname.
/N	Do not scan the user runtime library, URT.LB.
/O	Suppress error flags when bind overwrites occur.
/Q	Terminate binding after creating the files. filename.CK and filename.CM. You may edit these files, which contain the BIND command.
/SQUEEZE	Compress the runtime environment by eliminating code used for managing any overlayed subroutines. Use this switch only if you have not used the LINK line syntax to place the subroutines in an overlay. Note you can use this switch with ANSII standard or when you use the /V switch for overlaying.
/T=integer	Set the highest address in the shared partition. If integer is not a multiple of 2048 bytes, the binder rounds it down to the next lower 2048-byte multiple.
/Z=integer	Set the size of the stack for the default task.

Argument Switches

/B	Bind the externally referenced routines from this shared library into the root context.
/C	Use this module as a command file. CBIND requires this when defining overlays using square brackets.
/D	Load the nonshared code in this module as unshared data.
/H	Load the unshared code in this module as shared code.
/O	Allow overwrites in this module. See the /O CBIND switch.
/R	Issue a warning if any code in this module is not position-independent.
/S	Convert shared code modules to unshared code modules.
/U	Load local symbols from this module into the symbol file. /U works only if you applied it to this module in an earlier macroassembler command.
symbol/V=integer	Create this accumulating symbol and initialize it to integer.
integer/Z	Set the current ZREL base to the number specified by this argument.

Example

```
) CBIND/L=MYFILE.MP  MYFILE  UPDATSUB & )
&) HACKSUB )
```

Binds MYFILE, the main program, and two subprograms, UPDATSUB and HACKSUB. The binder output listing goes to MYFILE.MP.

CC *Language*

Compile one or more C source files (AOS/VS only).

Format

XEQ*[/S]* CC*[/global-switch-string]* source-file-data

or

CC*[/global-switch-string]* source-file-data

You can use either the XEQ CC command line or the CC.CLI macro provided to you to compile a C program module. We describe the CC.CLI macro later on.

For more information, refer to *The C Language Reference and Runtime Manual* (093-000264).

source-file-data contains exactly one main-filename, from zero to seven filename/include, any number of identifier/define, any number of identifier/define=value, and up to eight directory/search.

The optional /S switch on the XEQ command returns the program's termination message to STRING. If the program compiles successfully, STRING is empty. If the program does not compile, STRING contains an error message.

You may enter the components of source-file-data in any mixture and in any order. You may shorten XEQ to X, include to i, search to s, and define to d.

CC Switches

You may abbreviate switches up to ambiguity.

Many switches come in pairs /switch, /NOswitch, whose uses are mutually exclusive. One of the pair is the default. If you select neither member of the pair, the compiler assumes the default value.

/BACKSPACE*[=value]* Evaluate \b with the specified value. If you do not specify a value, the compiler evaluates \b as \10. If you do not use this switch, the compiler evaluates \b as \31.

/BRIEFERROR Direct the C compiler to print short error messages without printing the line on which the error occurred or the standard AOS/VS C header message on the error file. This latter message is normally 4-8 lines per compilation.

/CHECKOFFSET Check whether members of structures or unions with the same name are at the same offset, and of the same type, in all structures or unions.

/CODE Write out a code listing. You must use the /L switch to enable this switch.

/CONDITIONAL List all code, even code not conditionally compiled. This is the default.

/DEBUG Produce SWAT® information.

/DISALLOWANCIENT Do not evaluate assignment operators of the form =op. If you use this switch, the expression a = – 2 becomes a = (–2) rather than a = a – 2.

/E*[=filename]* Redirect the error listing to *filename*. If you use /E alone, the default error file is @OUTPUT. If you omit /E, the default error listing is @OUTPUT.

/ERRORCOUNT=n Set the maximum number of errors to list to n. If you omit a value, the default maximum is 100. If you exceed the maximum number of errors, compilation aborts.

/EXTERNMAP On external names, convert names with underscores (_) to corresponding names with question marks (?) in calling the macroassembler.

CC (continued)

/EXTL

Generate an external reference for each declaration with an explicit extern storage class declaration. Each reference must be satisfied in another module (as opposed to generating a partition definition block, which occupies the same storage as other partition definition blocks with the same name when the program is linked). This lets you pull default intializations from a library. You could also use the switch to reference entry points defined in assembly language. Any declaration preceded by the keyword extern, but not initialized, is assumed to be defined in some other module.

/FIXEDOVERFLOW

Turn on fixed overflow checking for all functions defined within the current compilation.

/FLAGDEFINES

Set a flag to tell whether the compiler used a #define macro on the current line. You must use the /L switch to enable this switch.

/FRAMESIZE

Print on the listing and error files the size in words that each function needs from the stack. This does not include space needed by the alloc, calloc, or malloc functions.

/INCDIRECTORY =directory-name

Specify the directory to prefix to an #include filename. If the file does not exist in the directory, the compiler then attempts to open the file from the current directory using the current searchlist.

/INCLUDES

Print include files on the listing file. You must use the /L switch to enable this switch. This is the default.

/L[=*filename*]

Produce a listing in *filename*. If you use /L alone, the default filename is @LIST, and if you omit /L, the default is not listing. You must use the /L switch to enable the /CODE, /FLAGDEFINES, /INCLUDES, /MAP, or /XREF switches.

/LINEID

Generate code to save the line number of the statement. Use the save number to report the location of runtime errors. Automatically set the /PROCID switch to identify functions and modules.

/LINK

This switch is not interpreted by the compiler, but rather by the CC.CLI macro. If you use this switch, it will call the CCL.CLI macro to link the various C modules together, providing there were no errors in the compilation. If used, the following CCL switches are available, and passed on the LINK:

/ALPHA
/HEAPSIZE=#
/LS
/MODMAP
/MODSYM
/MTOPS=#
/NOIOSWITCH
/NOSEA
/NOUNX
/NUMERIC
/RESERVE=#
/RT32
/SHAREDIO
/STACKSIZE=#
/TASKS=#

/LOWERCASE

Map identifiers to lowercase.

/LS

This switch is not a compiler switch, but is used by the CC.CLI macro, and creates a listing file of the form file.LS, where file is the name of the program being compiled, without the .C extension.

/MAP — Produce a map storage on the listing file. You must use the /L switch to enable this switch. This is the default.

/N — Do not produce an .OB file. If you omit this switch, you generate code.

/NOBRIEFERROR — Print out the standard AOS/VS C header message on the error file, print lengthy error messages, and attempt to pinpoint the token where the error occurred. This is the default.

/NOCHECKOFFSET — Do not check members of structures or unions with the same name for offset or type. This is the default.

/NOCODE — Do not list generated code. This is the default.

/NOCONDITIONAL — Do not list code that is not conditionally compiled.

/NODEBUG — Omit generation of SWAT information. This is the default.

/NODISALLOWANCIENT — Treat assignment operators of the form =op as equivalent to operators of the form op=. This is the default.

/NOEXTERNMAP — Do not map underscores (_) to question marks (?) in external names. This is the default.

/NOEXTL — All definitions that are external, including definitions with an explicit extern declaration, generate a partition definition block. Link will resolve all of the partition definition blocks with the same name to the same address. This is the default.

/NOFIXEDOVERFLOW — Do not check for fixed overflow. This is the default.

/NOFLAGDEFINES — Do not set a flag for #define macros used on the line. This is the default.

/NOFRAMESIZE — Do not display the stack space needed. This is the default.

/NOINCLUDES — Do not print include files.

/NOLINEID — Do not produce code to save line numbers of statements. This is the default.

/NOLOWERCASE — Do not map identifiers to lowercase automatically. This is the default.

/NOMAP — Do not produce a storage map.

/NOOPTIMIZE — Do not optimize the code. If you use this switch, your optimization level is 0. This is the default.

/NOPOINTERCHECK — Do not check pointers for validity. This is the default.

/NOPROCID — Do not produce code to save the procedure name. This is the default.

/NOSHAREDSTRING — Place quoted strings in unshared memory, where they can be modified. This is the default.

/NOSHORTARITHMETIC — Perform implicit conversions before carrying out arithmetic operations: convert short and char to int, short unsigned to unsigned, and float to double. This is the default.

/NOSIGNEDCHAR — Treat char variables as unsigned numbers in the range 0...255. This is the default.

/NOSTATISTICS — Do not generate compiler statistics. This is the default.

/NOSUBCHECK — Do not check for out-of-bounds array subscripts. This is the default.

CC (continued)

/NOUNDECLARED — Give an error message if a structure or union is not defined in the declaration. This is the default.

/NOWARNINGS — Do not generate any warnings. This is the default.

/NOXREF — Do not generate a cross reference. This is the default.

/O=filename — Redirect the object file to filename.OB. If you omit this switch, your object file is main-filename.OB.

/OPTIMIZE=n — Set the optimization level to n (n = 0, 1, 2, 3). If you omit a value, the default level is 3. If you omit the switch, the default level is 0.

Level Optimization

0 Do not optimize.

1 Eliminate unreachable code, such as an unlabeled statement following an uncoditional goto.

2 Include level 1 optimizations. In addition, eliminate redundant computations.

3 Include level 2 optimizations. In addition, remove invariant computations from do loops.

/PARTCODE=partition — Specify which partition shared code goes in. If you omit this switch, the compiler uses the default shared code partition. For more details, see the *AOS/VS Link and Library File Editor User's Manual.*

/PARTSTATIC=partition — Specify which partition the static data goes in. If you omit this switch, the compiler uses the default unshared data partition. For more details, see the *AOS/VS Link and Library File Editor User's Manual.*

/PARTSHORT=partition — Specify which partition the short static data goes in. If you omit this switch, the compiler will use the UD.SHORT_STATIC partition. For more details, see the *AOS/VS Link and Library File Editor User's Manual.*

/POINTERCHECK — Check all pointers for validity, and generate a stack traceback if the pointer is invalid. If a byte pointer (bit 0 = 0) occurs where a word pointer is expected, or a word pointer (bit 0 = 1) occurs where a byte pointer is expected, convert the pointer and continue execution. (The /LINEID switch will tell you the line number of the statement that uses the invalid pointer. Pointers are also checked at conversion time (byte vs. word) for validity, except that the NULL pointer is not given an error in this case.) There are some address traps that the /POINTERCHECK switch does not catch.

For most usages, the switch should help to identify places where byte and word pointers are used indiscriminately. To use this switch, your program must run in rings 4–7.

/PREPROCESSOR[=*file*] — Expand the preprocessor requests without running the rest of the compiler. This switch is useful primarily to find out what exactly is put out when a program has a complex series of #define macros, conditional compilation, and include files.

NOTE: If you invoke the compiler from MV/UX, this output file will not work across a pipeline.

When you use the /PREPROCESSOR switch, you can only use the following switches in addition to it:

/BRIEFERROR
/E[=file]
/INCDIRECTORY
=directory
/LOWERCASE
/N
/UNIQUE=n

Also note that you should disable the #assert statements that are used within the PACKETS subdirectory if you use the /PREPROCESSOR switch; otherwise, the rest of the compiler will not be invoked to give the size of the various structures. To disable these assertions, define the macro NASSERT before including the include files, or on the command line.

/PROCID — Produce code to save the procedure name. Use the saved name to report the offending function or module in runtime errors.

/REVISION=maj_rev*[.min_rev[.update[.pass[.]]]]* — Specify the revision number for the module. Each component of the revision number must be less than 256. If you omit this switch, the default is the compiler revision number.

/SHAREDSTRING — Place quoted strings in shared (write-protected) memory. This may give a slight performance gain when running multiple versions of the same program file, and slightly smaller code.

/SHORTARITHMETIC — Use narrow (16-bit integer) arithmetic for types short and short unsigned. Use single-precision (32-bit floating) arithmetic for the type float. This switch produces shorter or faster code for the given types, but accuracy suffers.

/SHORTSTATIC=n — If the number of words in static or external data items is less than equal to n, use the short addressing mode instead of the long addressing mode (that is, use an X-type instruction rather than an L-type instruction, and locate the data within the first 32K of memory).

/SIGNEDCHAR — Treat char variables as signed numbers in the range −128...127 rather than as unsigned numbers in the range 0...255. The compiler generates a warning for values outside the range −128...127. (Unsigned declarations such as unsigned char a; use values in the range 0...255.)

/STATISTICS — Generate compiler statistics, such as number of lines compiled.

/SUBCHECK — Check for out-of-bounds array subscripts. (You will not be able to use this switch if your program uses pointers instead of arrays.)

/TEMPDIR=directory — Specify an alternate directory for temporary files. If you omit this switch, the default is the current working directory.

/UNDECLARED — Do not give an error message if a structure or union is undefined. Since an undefined structure occupies no storage, you will not be able to use the structure in any expression, or other structure definitions. Other compilers might not check for undefined structures until the structure is referenced. This switch disables the check so that existing C programs can run without errors if they use this feature.

/UNIQUE=n — Set the number of unique characters in identifiers (from 8 to 32). If you omit this switch, the default number of unique characters is 32.

CC (continued)

/WARNINGS — List warnings.

/XREF — Generate a cross reference: append an alphabetical cross-reference of all program entities, with line numbers, to the listing file. You must use the /L switch to enable this switch. The cross listing includes #define macros. If any level 3 compilation errors occur, the compiler does not generate any cross reference, since they run off intermediate code.

Argument Switches

/DEFINE — Set up a #define from the command line.

/INCLUDE — Set up an include file from the command line.

/SEARCH — Search this directory for include files before a normal search following AOS/VS rules (current working directory and searchlist). If, within your source code, you specify the include file with quotation marks (" ") instead of angle brackets (< >), the directory containing the main file is searched first. You may specify up to 8 directories for searching.

/UNDEFINE — Remove any initial definition of #define macros that are automatically defined at the start of compilation. These macros are the following:

Macro	Definition
AOSVS	1
aosvs	1
LINE	Replaced by current line number.
PAGE	Replaced by current page number or −1 (# line).
FILE	Replaced by a string containing the filename.
REV	Replaced by a string containing the revision number.

The compiler accepts any number of local arguments with the /UNDEFINE switch appended.

CHAIN *Command*

Overwrite your CLI with the program named in pathname.

Format

CHAIN pathname *[argument-to-new-program]...*

Chain places the arguments in the initial interprocess communication (IPC) message to the new process. The new process can access these arguments through the ?GTMES system call. See Appendix B for details. The CLI is a 16-bit program and can chain to a 32-bit process or to another 16-bit process.

The CLI first tries to chain to pathname.PR. If that fails, the CLI chains to pathname.

WARNING: Chaining overwrites your CLI in main memory. The CLI will not return unless the chained program invokes it via the system call ?CHAIN.

Command Switches

/1= {IGNORE | WARNING | ERROR | ABORT} — Set CLASS1 to the specified severity level for this command.

/2= {IGNORE | WARNING | ERROR | ABORT} — Set CLASS2 to the specified severity level for this command.

/D — Enter the Debugger.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/Q — Set SQUEEZE to ON for this command.

Argument Switches

Use any argument switches appropriate for the program specified in pathname.

Example

) CHAIN MYPROG ↓

Load MYPROG into memory and begin execution. Do not create a new process, simply change this process's program.

CHARACTERISTICS *Command*

Set or display device characteristics.

Format

CHARACTERISTICS *[device]...*

If you do not supply an argument, your terminal becomes the default device. Set or display the device characteristics for a character device. Device characteristics control the way the device interprets input or sends output. The characteristics you set will be in effect *until you change them or log off the system, or POP to a previous level which has different characteristics.* You can issue successive CHARACTERISTICS commands.

Command Switches

/1= { IGNORE | WARNING | ERROR | ABORT }

Set CLASS1 to the specified severity level for this command.

/2= { IGNORE | WARNING | ERROR | ABORT }

Set CLASS2 to the specified severity level for this command.

/8BT

Tells the CLI to interpret all 8 bits of an ASCII character as data. The following octal codes will echo an up arrow followed by an alphabetic character regardless of whether this switch is set or not:

1 to 10
13
16 to 32
34 to 37.

/BREAK=value

(AOS/VS only) Sets the break function key to **value**. **value** can be one of the following:

BMOB Clears binary mode on break (default). Restores normal CTRL character handling.

CHARACTERISTICS (continued)

CAOB Causes a CTRL-C CTRL-A sequence on break (not echoed on the terminal).

CBOB Causes a CTRL-B CTRL-B sequence on break (not echoed on the terminal). Aborts the issuing process.

CFOB Causes a CTRL-C CTRL-F sequence on break. (Enables your program to detect a break.)

DCOB Disconnects the user process on the console and logs user off. Causes a modem to disconnect on break.

For more on /BREAK, see the How to Generate and Run AOS/VS manual.

/BAUD=n — Set a console's BAUD rate (decimal) (IAC and USAM systems only). Valid baud rates are

45.5	1800
50	2000
75	2400
110	3600
134.5	4800
150	7200
300	9600
600	19200
1200	38400

/CHARLEN=n — Set the character length (IAC and USAM systems only).

/CPL=n — Characters per line, in decimal.

/CRTn — Identifies nonstandard CRT types, where n is 4, 5 or a number from 7 to 15.

/DEFAULT — Used alone, this switch displays the default characteristics of a device. Used with other switches, it sets the default characteristics of a device. The only process authorized to set default characteristics is PID 2.

/EB0 — For echoing to occur on your terminal, you must set /EB0 and or /EB1. /EB0 echos control characters such as ↑A, ↑B, etc. It echos ESC as $. For more information, see ?GCHR in the appropriate programmer's manual (AOS or AOS/VS).

/EB1 — Echo characters exactly as they are input. For more information, see ?GCHR in the appropriate programmer's manual (AOS or AOS/VS).

/EOL — If you exceed the CPL line length on output, do not output a NEW LINE.

/EPI — Accept only even parity on input; if this switch is OFF, accept any parity on input.

/ESC — ESC character produces CA interrupt.

/FF — Output a form feed on open.

/FKT — Permit function keys to serve as delimiters in data-sensitive read operations. (WARNING: Do not use function keys to end CLI commands).

/IFC — Enable data flow control on input to and from a terminal.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/LPP=n — Lines per page, in decimal.

/LT	(AOS only) Output 60 (decimal) nulls on open and close.
/MOD	Device is on a modem interface.
/MRI	Monitor Ring Indicator.
/NAS	If this switch is ON, set non-ANSI standard bit. This switch tells the system the device is a non-ANSI standard device. On input, this switch converts carriage return to NEW LINE and line feed to carriage return. On output, it converts line feed to carriage return-line feed.
/NNL	Do not automatically append NEW LINES to card images.
/NRM	Do not allow this terminal to receive SEND messages Note that this does not suppress messages sent from the master CLI, PID 2.
/OFC	Enable data flow control on output to and from a terminal.
/OFF	Clear the bit in the device characteristics words for each of the command switches that follow, until the /ON switch or a delimiter occurs.
/ON	Set the following characteristics ON until the /OFF switch or a delimiter occurs. This bit is automatically set unless you include the /OFF switch. Therefore, this switch is optional.
/OTT	On input, convert octal 175 and 176 to octal 33.
/PARITY=n	Set the parity rate to odd, even, or none (IAC and USAM systems only).
/PBN	Packed format on binary read, 4 columns are put in 3 words; if this switch is OFF, memory right justifies columns (card readers only).
/PM	Page mode: if this switch is ON, write LPP lines per page on output, then suspend output until the user types CTRL-Q.
/PREVIOUS	Set the current characteristics of a device to the previous environment's characteristics (no arguments or other switches allowed).
/Q	Set SQUEEZE to ON for this command.
/RAC	If this switch is ON, send 2 rubouts after each NEW LINE and carriage return.
/RAF	If this switch is ON, send 21 (decimal) rubouts after each form feed.
/RAT	If this switch is ON, send 2 rubouts after each tab (CTRL-I).
/RESET	Set the characteristics of a device to its default characteristics. You must use this switch alone.
/SFF	If this switch is ON, simulate form feed.
/SPO	Output characters in even parity; if this switch is OFF, output characters as sent by the program.
/ST	Simulate a tab stop every 8th column.
/STOPBITS=n	Set the number of stop bits. Valid values are 1, 1.5, or 2 (IAC and USAM systems only).
/TO	Enable time-outs.
/TSP	Include trailing spaces; if this switch is OFF, suppress trailing spaces (card readers only).
/UCO	On output, convert lowercase to uppercase.
/ULC	On input, accept both upper- and lowercase; if this switch is OFF, convert lowercase input to uppercase.

CHARACTERISTICS (continued)

/WRP	Hardware generates NEW LINE on line-too-long.

You can identify your terminal with any of the following switches. The system also displays these to identify your terminal:

/HARDCOPY	Hard-copy terminals
/4010I	DGC Model 4010I
/6012	DGC Model 6012
/605x	DGC Model 6052 or 6053
/6130	DGC Model 6130
/CRT4	Other video display terminals

Argument Switches

None

Example

```
) CHARACTERISTICS ↵
/HARDCOPY/LPP=24/CPL=80/CRT=6012
/ON/ST/SPO/EB0/ULC/WRP
/OFF/SFF/EPI/8BT/RAF/RAT/RAC/NAS/OTT/EOL
/UCO/LT/FF/EB1/PM/NRM/MOD/TO/TSP/PBN
/ESC/FKT/NNL
)
```

Display the characteristics of the terminal; in this case the terminal is a hard-copy terminal.

```
) CHARACTERISTICS/LPP=24 ↵
)
```

Set the number of lines per page to 24 for your terminal.

```
) CHARACTERISTICS/PM/OFF/EPI ↵
)
```

Set page mode ON and accept both even and odd parity on subsequent input to the terminal.

```
) CHARACTERISTICS/CPL=132 @LPA ↵
)
```

Set the characters per line for the line printer to 132 decimal. To set characteristics, use the device name — in this case @LPA — rather than a queue name (e.g., @LPT).

CHECKTERMS *Command*

Check for the termination of a son process.

Format

CHECKTERMS

This command displays the process termination message from any son processes. If a process terminates abnormally (e.g., console interrupt, trap), the CLI outputs an appropriate message. See the appropriate programmer's manual (AOS or AOS/VS) for a discussion of ?RETURN.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by **pathname** instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) PROCESS PROG1 ↵
PID: 14
) TERMINATE 14 ↵
) CHECKTERMS ↵
PROCESS TERMINATION, PID: 14
*ABORT*
TERMINATED BY A SUPERIOR PROCESS
)
```

The first command creates a subordinate swappable process with program PROG1. The second command terminates process 14. The CHECKTERMS command checks PID 14's termination message.

CLASS1 *Command*

Set or display CLASS1 setting.

Format

CLASS1 *[severity-level]*

We describe CLI exceptional condition handling in Chapter 4.

The following are severity-levels:

IGNORE	The CLI displays no message; it continues processing your input as best it can.
WARNING	The CLI displays a warning message and continues processing your input as best it can.
ERROR	The CLI displays an error message and discards the input that is in the command buffer at the time it encounters the mistake. The command buffer contains all input from the last prompt to the NEW LINE character. This may be one command, multiple commands, or a macro.
ABORT	Your process terminates at once.

NOTE: When you log on, the default setting for a CLASS1 mistake is ERROR. In batch, CLASS1 is set to ABORT by default.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set CLASS1 severity level to the previous environment's severity level (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) CLASS1 ↓
ERROR
) CLASS1 ABORT ↓
)
```

First, displays the current CLASS1 setting, then changes it to ABORT.

CLASS2 *Command*

Set or display CLASS2 setting.

Format

CLASS2 *[severity-level]*

We describe CLI exceptional condition handling in Chapter 4.

The following are severity-levels:

IGNORE	The CLI displays no message; it continues processing your input as best it can.
WARNING	The CLI displays a warning message and continues processing your input as best it can.
ERROR	The CLI displays an error message and discards the input that is in the command buffer at the time it encounters the mistake. In this instance the command buffer contains all input from the last prompt to the NEW LINE character. This may be one command, multiple commands, or a macro.
ABORT	Your process terminates at once.

NOTE: When you log on, the default setting for CLASS2 is WARNING.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set CLASS2 severity level to the previous environment's CLASS2 severity level (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) CLASS2 ↓
WARNING
) CLASS2 IGNORE ↓
)
```

First, displays the current CLASS2 setting, then changes it to IGNORE.

CLINK *Language*

Link object modules to form an executable COBOL program (AOS/VS only).

Format

CLINK main-objectmodule
[subprogram-objectmodule]...

CLINK is a macro that invokes the AOS/VS Link utility to make COBOL object modules into an executable program. For a list of the switches that the macro accepts, see the *AOS/VS Link and Library File Editor (LFE) User's Manual*(093-000245).

COBOL *Language*

Compile a COBOL source file.

Format

For AOS:

XEQ COBOL source-pathname *[listfile/L]*
[objectfile/R]

or

COBOL source-pathname *[listfile/L])*
[objectfile/R]

For AOS/VS:

XEQ COBOL source-pathname

or

COBOL source-pathname

You can use either the XEQ COBOL command line or the COBOL.CLI macro provided to you to compile a COBOL program. COBOL is a macro that you use to compile a COBOL source file.

source-pathname specifies the source program file you want compiled. *listfile* specifies the file or device to which you want the listing file output. This may be the console (@OUTPUT), the line printer (@LIST), or a disk or tape file. *objectfile* specifies the name the compiler assigns to the object file. By default, the compiler names the object file source-pathname.OB.

For a complete discussion of the COBOL programming language and the CLI COBOL command line, see the *COBOL Reference Manual (AOS)*(093-000223) or the *COBOL Reference Manual (AOS/VS)*(093-000289).

Below are two sets of COBOL switches, one for AOS and one for AOS/VS. Only a few switches are common to both sets. The argument switches are valid only for AOS.

COBOL Switches (AOS only)

/A Produce an address map of the relative locations of the Procedure Division lines.

/C Source code is in card format. By default, the compiler assumes the source is in text format.

/D Compile debug lines and load code for the interactive debugger.

/E Compile language extensions. Use this switch if you want octal values produced for alphanumeric literals (this conflicts with ANSI standard COBOL features).

COBOL (continued)

/G List the generated machine code. This switch overrides /A.

/L List source code at the current list file.

/M Produce a map of data and procedure storage in the object file.

/P Do not generate an object file.

/Q Do not compile, but only scan the source file code and produce a cross-reference table.

/S List compilation statistics (e.g., number of lines, speed of compilation).

/V Compile for virtual code. This switch provides automatic segmentation of procedure division calls.

/W Suppress warning messages.

/X Include a cross-reference table in the listing file.

COBOL Switches (AOS/VS only)

/ANSI Various ANSI standards override the usual AOS/VS COBOL data-manipulation methods.

/CARD Source code is in card format. By default, the compiler assumes the source is in text format.

/CODE or /C Print a generated code listing on the list file. This switch overrides /CODEMAP. /L must accompany the /CODE switch.

/CODEMAP Print a code offset map on the list file. /L must accompany this switch. If both /CODE and /CODEMAP appear in the same command line, the compiler ignores /CODEMAP.

/D Compile debug lines.

/DEBUG Output symbol and line information for use by the SWAT debugger.

/E[=*pathname*] If you don't specify a pathname, write error messages to @OUTPUT. If you specify a pathname write error messages to that file.

/ERRORCOUNT =integer Terminate compilation after the specified number of errors. The default value is 100.

/FRAMESIZE Produce a table that indicates the minimum stack space required to run a COBOL program. Sometimes this switch will also give you the preferred amount of stack space along with the minimum amount. The output from this switch goes to @OUTPUT.

/L Write the listing to the current list file. The listing consists of line-numbered source text, a storage map for all variables (unless you specify /NOMAP), and any compilation error messages. For additional information, specify the appropriate switches in addition to /L: for example, /L and /XREF produce a cross-reference listing.

/L=pathname Write the listing to the file specified by pathname. For details about the listing, see the /L switch.

/LEF Do not suppress generation of LEF instructions. This is the default.

/LINEID Generate code to keep track of source line numbers at execution time and to print the line number if a fatal error occurs. /LINEID includes the function of /PROCID.

/MAP Include a storage map in the listing. This is the default. /L must accompany this switch.

/MAPCASE Translate all identifiers into uppercase before compilation.

/N Suppress production of the object file.

/NOCODE Do not print a generated code listing on the list file.

/NOCODEMAP Do not print a code offset map on the list file.

/NOCOPIES Suppress printing of all copy files.

/NODEBUG Do not output symbol and line information.

/NOFRAMESIZE	Do not produce a table that indicates the minimum stack space required to run a COBOL program. This is the default.
/NOLEF	Do not generate LEF instructions.
/NOLINEID	Do not generate code to keep track of source file line numbers at runtime. This is the default.
/NOMAP	Suppress printing of the storage map.
/NOMAPCASE	Do not translate identifiers into uppercase. This is the default.
/NOPROCID	Do not generate code to keep track of procedure names at runtime. This is the default.
/NOREL	Commands the compiler to use revision 2.18 and below to process relative files.
/NOSTATISTICS	Do not print compilation statistics. This is the default.
/NOSUBCHECK	Do not generate code to check out-of-bound subscripts at runtime. This is the default.
/NOTES	Print severity 0 error message.
/NOWARNINGS	Suppress severity 1 error messages.
/NOX	Perform extended arithmetic operations in nonextended mode.
/NOXREF	Do not generate a symbol cross-reference table. This is the default.
/O=pathname	Write the object file to pathname.OB.
/OPTIMIZE *[=1 or 2 or 3]*	Set compiler optimization to the specified level, ranging from 1 (lowest) to 3 (highest). If you include this switch but do not specify a level, the default level is 3.
/PROCID	Save the procedure names at runtime, and print the procedure name if a fatal error occurs.
/STATISTICS	Write compilation statistics to @OUTPUT.
/SUBCHECK	Compile code into program to check for out-of-bounds subscripts.
/TMPDIR=string	Add string as a prefix to the beginning of all temporary filenames.
/WARNINGS	Do not suppress severity 1 error messages. This is the default.
/XREF	Include a cross-reference table in the listing file.

Argument Switches (AOS only)

If an argument specifies the list file or the object file, it must have the appropriate argument switch appended to it. The pathnames may appear in any order.

/L List the source code at listfile. The CLI assumes @LIST if you do not use this switch.

/R Produce the object file at objectfile. If you do not use this switch, the compiler uses the source file's name with an .OB extension.

Example

For AOS:

```
) COBOL/L/X/W FILE1 FILE1.LS/L )
```

Compile the source file FILE1 and produce an object file named FILE1.OB (the default name). The listing file FILE1.LS will contain a source listing (/L), a cross-reference table (/X), and error messages (automatically). The compiler will suppress warning messages (/W).

For AOS/VS:

```
) COBOL/L=FILE1.LS/E=FILE1.ER/DEBUG FILE1 )
```

Compile the source file FILE1 and produce an object file named FILE1.OB (the default name). The listing file FILE1.LS will contain a source listing (/L). Error messages are sent to FILE1.ER. Symbol and line information are generated for later use by the SWAT debugger (/DEBUG).

COMMENT *Command*

Include a comment.

Format

COMMENT *[text]*

This command allows you to insert a comment (text) within a CLI command file. The CLI ignores the text, which remains part of the file. The COMMENT command is useful primarily in CLI macros, to explain what is happening.

Within the text line, the CLI recognizes angle and square brackets, parentheses, and semicolons as in any command. It attempts to expand them as usual.

Command Switches

/1=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) CREATE/I FOO.CLI )
)) COMMENT This macro writes "Hello." )
)) WRITE Hello )
)) ) )
)
```

Creates a macro, with a comment.

CONNECT *Command*

Establish a Customer-Server connection.

Format

CONNECT {username:procname / process-ID}

This command directs the system to establish a connection between you and the server process that you specify. After making the connection you should monitor the server process with the CHECKTERMS command. If the server process terminates for any reason, then you must disconnect from the server. You can disconnect by using the CLI command DISCONNECT. (See the appropriate programmer's manual for a complete description of the customer-server relationship).

Command Switches

/1=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/S		Store the server's process ID in the current STRING buffer.

Argument Switches

None

Example

```
) CONNECT OP:SRVR ↓
SERVER'S PID: 14
)
```

Connect the user's process with server process OP:SRVR which is PID 14.

```
) CONNECT/S 22
SERVER'S PID: 22
) STRING ↓
22
)
```

Connect the user's process to PID 22 and store the PID number in STRING. The /S switch is useful if you want to use the server's PID as an argument to a command.

[!CONSOLE] *Pseudo-Macro*

Expand to the console name.

Format

[!CONSOLE]

This pseudo-macro does not accept arguments.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE MY CONSOLE NAME IS [!CONSOLE] ↓
MY CONSOLE NAME IS CON12
)
```

NOTE: The !CONSOLE pseudo-macro will only return the console number of processes under the control of EXEC. For batch processes, !CONSOLE will return a null string.

CONTROL *Command*

Send a control message to a process.

Format

CONTROL ipcport argument *[argument]...*

argument is a message string sent as an Interprocess Communication (IPC) to the process being controlled. The system operator normally uses this command to control the EXEC, INFOS, or XODIAC. You can use CONTROL to control user programs if you've written the program to receive IPCs. See the appropriate programmer's manual for more information about IPCs.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/I		Messages from @INPUT follow on successive lines. The system sends each line as a separate IPC. The messages end when you type a line containing a single) .
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/M		This macro file contains the messages. The system sends each line of the macro as a separate IPC. The macro file ends on a line containing a single) .
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) CONTROL @EXEC RESTART @LPB )
)
```

Directs the spooler to restart output on the line printer.

```
) CONTROL @EXEC ENABLE @CON23 )
)
```

Directs the EXEC process to enable @CON23.

CONVERT *Utility*

Convert an RDOS .RB file to an AOS or AOS/VS .OB file.

Format

XEQ CONVERT pathname

RDOS is another Data General operating system. It supports an .RB relocatable binary module, which is not compatible with AOS or AOS/VS.

The CONVERT utility can convert an RDOS .RB relocatable binary file to an AOS or AOS/VS object file. The command line takes one argument, the input **pathname** (you can omit the .RB extension). CONVERT does not modify the RDOS file; it creates an AOS or AOS/VS object file with the same name but with the .OB extension.

CONVERT Switches

None

Argument Switches

None

Example

```
) XEQ CONVERT PLUS24 )
PLUS24.RB
)
```

Produces an AOS or AOS/VS object file named PLUS24.OB from an RDOS object file named PLUS24.RB in the working directory. (The CONVERT program displays the message *PLUS24.RB* when it opens the input file.)

COPY *Command*

Copy one or more files to a destination file.

Format

COPY **dest-file sourcefile** *[sourcefile]...*

If the destination file does not already exist, then its specifications depend on the first (or only) **sourcefile**. If the first **sourcefile** is a disk file, then **dest-file** will have the same specifications as **sourcefile**. If the first **sourcefile** is a peripheral device, then **dest-file's** default specifications are as follows:

File type	User Data File
Record type	Unspecified. You must specify the record type when you open the file, or defer it until you read or write the file
Control parameters	None
Element size	512 bytes (under AOS/VS, you can specify a default element size during system generation)
Maximum index levels	3
Time block	Time of creation, time of last access, and time of last modification are set to the current time

If the destination file already exists, then you must use either the /A or /D command switch. /A appends the contents of the **sourcefile(s)** to **dest-file**; and /D deletes **dest-file**, then creates a new file with the same name and specifications and copies the **sourcefile(s)** into it. If any one of the sourcefiles does not exist, a warning is issued and processing of the copy stops.

NOTE: The COPY command does not copy the User Data Area (UDA) of a file.

COPY (continued)

Command Switches

/1= (IGNORE | WARNING | ERROR | ABORT) — Set CLASS1 to the specified severity level for this command.

/2= (IGNORE | WARNING | ERROR | ABORT) — Set CLASS2 to the specified severity level for this command.

/A — Append new data to the existing data in dest-file.

/B — Binary mode (for character devices); no interpretation or translation of special characters.

/BACKUP — Use checkpointing when copying files with FTA. If the file transfer is aborted, a UID will be returned for use with /BACKUP=UID.

/BACKUP=UID — Use a UID returned by FTA for recovering from an aborted file transfer for which the /BACKUP switch was specified.

/COMPRESS — Use compression when copying files with FTA.

/D — Delete dest-file (it must exist) and recreate dest-file using the same specifications as for the old dest-file.

/FTA — Use Xodiac FTA for copying files.

Note that only the /A, /D, /V, /COMPRESS, and /BACKUP switches are valid with /FTA.

/IDENSITY=mode — Control the magnetic tape density for input files. Use this switch with variable-density units, (like MTBs). These are the mode options:

Mode	Density
800	800 BPI
1600	1600 BPI
6250	6250 BPI (AOS/VS only)
ADM	Automatic Density Matching

/IMTRSIZE=block-size (bytes) — Control the magnetic tape block size for input files.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/ODENSITY=mode — Control the magnetic tape density of output files. Use this switch with variable-density units, (like MTBs). These are your mode options:

Mode	Density
800	800 BPI
1600	1600 BPI
ADM	Automatic Density Matching

/OMTRSIZE=block-size (bytes) — Control the magnetic tape block size for output files.

/Q — Set SQUEEZE to ON for this command.

/V — Display the pathname of the sourcefile copied.

Argument Switches

None

Example

```
) COPY OUTPUTFILE FILEA ↵
)
```

Copies FILEA to OUTPUTFILE; creates OUTPUTFILE using FILEA's specifications.

```
) COPY/A TESTALL TEST1 TEST2 TEST3 ↓
)
```

Appends TEST1, TEST2, and TEST3 to the end of TESTALL.

```
) COPY TESTA @MTB0:0 ↓
)
```

Copies file on MTB0:0 to TESTA and uses default specifications to create TESTA.

```
) COPY/V TEST1 TEST2 ↓
TEST2
)
```

Copies TEST2 to TEST1, using the /V switch to display the source file's pathname.

```
) COPY/FTA :NET:M6000_4:UDD:BJ:OUTPUTFILE &↓
&)□INPUTFILE ↓
)
```

Copies INPUTFILE to the remote system; creates OUTPUTFILE using INPUTFILE's specifications.

```
) COPY/V XFILE @DPM0 ↓
DPM0
)
```

Insert the source diskette in the unit. (XFILE is the disk filename chosen for the copy. @DPM0 is the devicename. The devicename for a 737,000-byte diskette unit under AOS/VS is @DPJ10.)

Replace the source diskette with the destination diskette. This diskette must be hardware formatted (as supplied by DG), but it need not be software formatted with the Disk Formatter.

```
) COPY/V @DPM0 XFILE ↓
XFILE
) DELETE/V XFILE ↓
DELETED XFILE
)
```

(Delete copy file to free disk space.)

The copy is complete. Remove the copy diskette. This procedure copies a physical diskette — and its entire file structure — to another diskette. It can copy any diskette that's been hardware formatted for DG hardware, regardless of the file structure or software format on the diskette.

CPUID *Command*

Display the CPU identification (AOS/VS only).

Format

```
CPUID
```

This command displays your computer's central processing unit (CPU) identification. For an explanation of this number, consult *How to Generate and Run AOS/VS*.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
)CPUID ↓
CPUID 4437000407
)
```

Displays the CPU identification.

CREATE *Command*

Create a file.

Format

CREATE pathname *[resolution-pathname]*

If you omit switches for this command, the system creates a text file that you can use for text or source code. When you CREATE a link entry to a file, pathname is the linkname you'll use to access the resolution file, and *resolution-pathname* is the resolution file's pathname.

Command Switches

/1= {IGNORE | WARNING | ERROR | ABORT}
Set CLASS1 to the specified severity level for this command.

/2= {IGNORE | WARNING | ERROR | ABORT}
Set CLASS2 to the specified severity level for this command.

/DATASENSITIVE
Create the file with data-sensitive record format.

/DIRECTORY
Create a directory. If you use the /MAXSIZE= switch also, create a control point directory (type CPD) with the specified maximum size.

/DYNAMIC
Create the file with dynamic record format.

/ELEMENTSIZE=n
Set the file element size to the specified value. A file element is a set of contiguous 512-byte blocks. This switch lets you create files with a lot of continguous space. Under AOS, the default size is 1 block. Under AOS/VS, the default size is 4 contiguous blocks, or the value chosen at system generation.

/FIXED=n
Create the file with the specified fixed-length record format.

/HASHFRAMESIZE=n
Set the hash frame size for this directory or control point directory. The default hash frame size is 7, which suits directories that contain about 140 files. For optimum access, if a directory will contain many more or fewer files, divide the number of files by 20 and take the nearest prime number. For example, the best hash frame size for 300 files is 17.

/I
Take the contents of the file from subsequent lines of the @INPUT file. The last line must contain a single) .

/INDEXLEVELS=n
Set the maximum number of index levels to the specified value. The default is 3. The system starts with 0 index levels and creates others as needed, up to 3 levels. An index level of 0 limits a file to 1 (contiguous) element; a level of 1 limits it to 128 elements; a level of 2 limits it to 1282 elements; and so on.

/L
Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname
Write CLI output to the file specified by pathname instead of to @OUTPUT.

/LINK
Create a link, named in pathname, to the resolution pathname specified as the second argument.

/M
Take the contents of the file from subsequent lines of the current macro body. The last line of the macro body unit must contain a single) .

/MAXSIZE=n
Set the maximum size for a control point directory.

/Q
Set SQUEEZE to ON for this command.

/TYPE=type — Create a file of type type.

Table 2-6 contains the list of valid file types.

type can be in the following forms:

XXX 3-letter mnemonic

n decimal number (64–255)

/VARIABLE — Create the file with variable record format.

Argument Switches

None

Example

```
) CREATE/I PROG.FR ↓
))DIMENSION ARRAY(100) ↓
.
.
.
))END ↓
))) ↓
)
```

Creates a FORTRAN IV source file named PROG.FR.

```
) CREATE/LINK LNAME :UDD:USERNAME:FILE1 ↓
)
```

Creates a link file containing a complete pathname to FILE1.

```
) CREATE/DIRECTORY PROJECT1 ↓
)
```

Creates a directory called PROJECT1.

CURRENT *Command*

Display the current CLI environment's settings.

Format

CURRENT

This command displays the current settings for CHARACTERISTICS, CLASS1, CLASS2, DATAFILE, DEFACL, DIRECTORY, LEVEL, LISTFILE, LOGFILE, PROMPT, SCREENEDIT, SEARCHLIST, SQUEEZE, STRING, SUPERPROCESS, SUPERUSER, TRACE, and VAR0–VAR9.

Command Switches

/1= {IGNORE | WARNING | ERROR | ABORT} — Set CLASS1 to the specified severity level for this command.

/2= {IGNORE | WARNING | ERROR | ABORT} — Set CLASS2 to the specified severity level for this command.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/Q — Set SQUEEZE to ON for this command.

CURRENT (continued)

Argument Switches

None

Example

```
) CURRENT )
LEVEL           0
SUPERUSER       OFF
SUPERPROCESS    OFF
SCREENEDIT      OFF
SQUEEZE         OFF
CLASS1          WARNING
CLASS2          ERROR
TRACE
VARIABLES       0  0  0  0  0
                0  0  0  0  0
LISTFILE        @LIST
DATAFILE        @DATA
LOGFILE
DIRECTORY       :UDD:JOHN
SEARCHLIST      :UDD:JOHN,:PER
DEFACL          JOHN,OWARE
STRING
PROMPT
CHARACTERISTICS
                /605X/LPP=24/CPL=80/BAUD=9600
                /PARITY=none/CHARLEN=8
                /STOPBITS=1/BREAK=BMOB
                /ON/ST/EB0/ULC/WRP
                /OFF/SFF/EPI/8BT/SPO/RAF/RAT
                /RAC/NAS/OTT/EOL/UCO/MRI/FF
                /EB1/PM/NRM/MOD/TO/TSP/PBN
                /ESC/FKT/NNL/SHR/OFC/IFC
)
```

Displays the current CLI environment's settings on an AOS/VS system. There are fewer characteristics in a CLI environment on an AOS system.

DATAFILE *Command*

Set or display the current DATAFILE pathname.

Format

DATAFILE *[pathname]*

This command sets the DATAFILE to *pathname*. The CLI passes DATAFILE to any process created by an EXECUTE, XEQ, or DEBUG command. The CLI itself does not use the DATAFILE. When coding a program that must open and use a data file, you can use the generic filename (@DATA) within your program instead of the specific name. Then, before you execute the program, you can use the DATAFILE command to select the data file you want the program to use.

Command Switches

/1=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS2 to the specified severity level for this command.
/G		Set the filename to @DATA (no arguments allowed).
/K		Set to null string (no arguments allowed).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set DATAFILE to previous environment's DATAFILE (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) DATAFILE ↓
@DATA
) DATAFILE MYFILE ↓
) DATAFILE
:UDD:CHRIS:MYFILE
)
```

First, displays the current DATAFILE, then set it to MYFILE.

[!DATAFILE] *Pseudo-Macro*

Expand to the pathname of the current DATAFILE.

Format

[!DATAFILE]

This pseudo-macro does not accept arguments.

Macroname Switches

/P Expand to previous environment's data file pathname.

Argument Switches

None

Example

```
) WRITE THE CURRENT DATA FILE IS [!DATAFILE] ↓
THE CURRENT DATA FILE IS @DATA
) PUSH ↓
) DATAFILE :UDD:USER:WORK ↓
) WRITE NOW THE CURRENT DATA FILE IS & ↓
&) [!DATAFILE] ↓
NOW THE CURRENT DATA FILE IS
:UDD:USER:WORK
) WRITE [!DATAFILE/P] ↓
@DATA
)
```

First, evaluates [!DATAFILE] and writes the current data file pathname, which is the generic @DATA. Then changes environment, and sets a new data file for the new environment. Evaluates and writes [!DATAFILE] for the current environment, and then, using the /P switch, for the previous environment.

DATE *Command*

Set or display the current system date.

Format

DATE *[date]*

The date can be set only by the master CLI (process-ID 2). Use one of the following formats for date:

11 26 84

26-NOV-84

In the second format, you can use three character abbreviation for the month.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) DATE 11 26 84 ↵
) DATE ↵
26-NOV-84
)
```

Sets and displays the date.

```
) DATE 26-N-84 ↵
) DATE ↵
26-NOV-84
)
```

Set and display the date. (Note that you can use N because no other month begins with N).

[!DATE] *Pseudo-Macro*

Expand to the current system date.

Format

[!DATE]

This pseudo-macro does not accept arguments.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE TODAY IS [!DATE]. ↵
TODAY IS 26-NOV-84.
)
```

First expand to the system date and then output to the screen, "TODAY IS 26-NOV-84."

DEASSIGN *Command*

Deassign a previously assigned character device.

Format

DEASSIGN character-device *[character-device]...*

After you ASSIGN a device, you control it until you either DEASSIGN it or log off the system.

You may use templates in the character device argument.

Command Switches

/1=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS1 to the specified severity level for this command.
/2=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) DEASSIGN @MTB )
)
```

Deassigns the tape drive, that you previously assigned.

DEBUG *Command*

Execute the specified program and enter the Assembly Language Debugger.

Format

DEBUG pathname *[argument-to-new-program]...*

DEBUG creates a subordinate process that executes the program named in pathname (it must be a program file). The new program starts in the Debugger. The CLI places the arguments in the initial IPC message to the new process. The new process can access these arguments through the ?GTMES system call. See Appendix B for details.

The CLI first tries to debug pathname.PR. If this fails, the CLI tries pathname.

For more on DEBUG, see the *AOS Debugger and DiskFile Editor User's Manual* (093-000195) or the *AOS/VS Debugger and File Editor User's Manual* (093-000246).

Command Switches

/1=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS1 to the specified severity level for this command.
/2=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS2 to the specified severity level for this command.
/I		Create input for the program from @INPUT. The last line of input must contain a single) .
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/M		Create input for the program from the macro body. The last line of input must contain a single) .
/Q		Set SQUEEZE to ON for this command.
/S		Return the termination message to STRING.

DEBUG (continued)

Argument Switches

The *AOS Debugger and Disk File Editor User's Manual* (093-000195) or the *AOS/VS Debugger and File Editor User's Manual* (093-000246) explains DEBUG's argument switches

Example

```
) DEBUG MYPROGRAM ↵
AOS USER DEBUGGER, REV xx
# 0=000000 # 1=000000 # 2=000000 # 3=000000
.
.
+ BYE ↵
)
) DEBUG VSPROGRAM ↵
AOS/VS User Debugger- Rev. XX
00000000000 00000000000 00000000000 00000000000
00000000000
_$Z
)
```

[!DECIMAL] *Pseudo-Macro*

Convert an octal number to decimal.

Format

[!DECIMAL octal-number]

The number must be a positive octal integer in the range 0 to 37,777,777,777. The result will be in the range 0 to 4,294,967,295.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE 112 OCTAL = [!DECIMAL 112] DECIMAL. ↵
112 OCTAL = 74 DECIMAL.
```

DEDIT *Utility*

Edit disk file locations (AOS only).

Format

XEQ DEDIT pathname

DEDIT invokes the Disk File Editor utility, which allows you to examine and change the contents of disk file locations. The DEDIT utility uses a subset of AOS Debugger commands. It is functionally identical to the AOS Debugger, except that it cannot set breakpoints or examine accumulators. For more on DEDIT, consult the *AOS Debugger and Disk File Editor User's Manual* (093-000195). (To edit disk file locations under AOS/VS, see the FED utility.)

DEDIT Switches

/I=pathname	Take DEDIT commands from pathname. This lets you build a file of DEDIT commands and execute it with a single CLI command. You should terminate commands in the file normally; i.e., with either NEW LINE or carriage return. You could build the command file earlier, by using the /L switch. The file must end with a BYE command.
/L=pathname	Save all DEDIT commands in a file identified by pathname.
/S=pathname	Include the symbol table file identified by pathname.

Example

```
) XEQ DEDIT MYFILE.PR ⤸
+ START: 001762 ⤸
.
.
+ BYE ⤸
)
```

This sequence executes DEDIT on user program MYFILE in directory DIR1; DEDIT then prints its prompt (+) and accepts editing commands. At the session's end, the BYE command returns control to the CLI.

DEFACL *Command*

Set or display the default access control list.

Format

DEFACL *[username,access]...*

To display the current default access control list (ACL), enter DEFACL without arguments. The system default ACL is *username,* OWARE. When you create a new file, this is usually its ACL.

To change the current default ACL, enter the *username* and specify new *access* values.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/D		Return to the system default ACL (username, OWARE).
/K		Set the default ACL to no ACL (no arguments allowed). This denies everyone except Superusers access to your files until the ACL changes again.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set default ACL to the previous environment's default ACL (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

DEFACL (continued)

Argument Switches

None

Example

```
) DEFACL ↓
SMITH, OWARE
) DEFACL  SMITH,R ↓
) DEFACL ↓
SMITH,R
) DEFACL/D ↓
) DEFACL ↓
SMITH,OWARE
)
```

The first command displays the current default ACL, which happens to be the system default ACL. The second command changes the ACL to Read access only. The next command, displays the new default ACL. Then, the following command uses the /D switch to change the ACL back to the system default ACL. Finally, the last command displays the current default ACL once again.

[!DEFACL] *Pseudo-Macro*

Expand to the current user default access control list.

Format

[!DEFACL]

This pseudo-macro does not accept arguments.

Macroname Switches

/P Use the previous environment's default ACL.

Argument Switches

None

Example

```
) WRITE THE CURRENT USER DEFAULT ACL IS& ↓
&)[!DEFACL] ↓
THE CURRENT USER DEFAULT ACL IS
SMITH,OWARE
)
```

First, evaluates the pseudo-macro [!DEFACL], then writes the resulting argument list on the terminal.

DELETE *Command*

Delete one or more files.

Format

DELETE pathname *[pathname]...*

Deleting a directory deletes all files in the directory. You cannot delete a directory that contains inferior directories.

You may use templates in the pathname arguments.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/C		Confirm each deletion. The CLI displays each filename and waits for you to confirm the deletion. Enter Y to delete the file, N or NEW LINE not to delete the file
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/V		Verify each deletion with a list of the files deleted.

Argument Switches

None

NOTE: When a filename or pathname has an @ prefix, no deletion occurs. If the file exists, a normal return is taken. If the file does not exist, an error is returned.

Example

```
) DELETE/V ADAM ↵
DELETED ADAM
) DELETE/C TEST.- ↵
=TEST.01? Y ↵
=TEST.02? Y ↵
=TEST.03? ↵
FILE NOT DELETED
)
```

DGL *Utility*

Compile a DG/L™ source file.

Format

XEQ DGL source-pathname *[argument]...*

The DGL utility compiles a DG/L™ source file. The DGL command first looks for source-pathname.DG. If that fails, it looks for source-pathname. By default, executing the command produces an object file named source-pathname.OB, and produces no listing.

The optional arguments, with the appropriate switches appended, can specify the listing, error, and object files, or various code generation and selective compilation options.

The DGL command switch /CODE=symbol and the argument switch symbol/C let you generate code for a specified machine/operating system combination. You may generate AOS or RDOS code on AOS/VS, and RDOS code on AOS. The legal values for symbol are as follows (note that each machine/operating system combination has several symbols, and that some symbols designate more than one combination):

Symbol	Machine	Operating System
N	NOVA	RDOS
NOVA	NOVA	RDOS
E	ECLIPSE	RDOS
ECLIPSE	ECLIPSE	RDOS
RDOS	ECLIPSE	RDOS
A	ECLIPSE	AOS
AOS	ECLIPSE	AOS
16	ECLIPSE	AOS
X16	ECLIPSE	AOS
VS16	ECLIPSE	AOS
A	MV/family-16	AOS/VS
AOS	MV/family-16	AOS/VS
16	MV/family-16	AOS/VS
X16	MV/family-16	AOS/VS
VS16	MV/family-16	AOS/VS
X	MV/family-32	AOS/VS
X32	MV/family-32	AOS/VS
32	MV/family-32	AOS/VS
VS32	MV/family-32	AOS/VS

If you omit this switch, DG/L generates code for the current environment.

For a complete description of the DG/L programming language and the CLI DGL command line, see the *DG/L™ Runtime Library (AOS and AOS/VS) User's Manual* (093-000159) and the *DG/L Reference Manual* (093-00229).

DGL Switches

/A	Continue compilation past the phase that catches syntax errors, even if there are syntax errors. Without this switch, errors found in the syntax phase cause the compiler to skip the other phases.
/B	Produce a brief output listing (source text and storage map only).
/C	Check syntax of source text, but do not check semantics or generate code. Since syntax checking takes less time than a full compilation, this option is useful in early program development.
/CODE=symbol	Generate code for the specified machine/operating system combination. You may generate AOS or RDOS code on AOS/VS, and RDOS code on AOS. See the previous list for the legal values for symbol. If you omit this switch, DG/L generates code for the current environment.
/DEBUG	Produce DS and DL blocks in the object file for use by the SWAT debugger.
/E=pathname	Write error messages to the file specified by pathname instead of to @OUTPUT.
/F	Produce an error message if the compilation results. in an attempt to generate a floating-point instruction.
/G	Create local symbols out of line numbers, and global symbols out of procedure names. The debugger can use these symbols.

/H	(For NOVA® code only.) Generate code usable on a target machine that has hardware multiply/divide instructions (otherwise, DG/L uses software multiply/divide).
/I	Do not list the contents of INCLUDE files on the compilation listing.
/INNER	Produce code you want to link for an inner (4–6) ring, and that you can call through a gate array from an outer ring. This switch is valid only when /CODE= has the value X, X32, or VS32.
/L	Write a listing to the current list file.
/L=pathname	Write a listing to the file specified by pathname.
/M	Generate code that will run on a mapped ECLIPSE machine; i.e., the compiler can use short LEF instructions. The compiler assumes /M when it generates code for AOS systems, but not for ECLIPSE RDOS.
/N	Do not generate object code, but proceed otherwise with all phases of compilation.
/NOLEF	Do not use short LEF instructions in the code. This switch is the opposite of /M.
/O=pathname	Write the object file to pathname.
/OPT=string	Perform conditional compilation; i.e., compile lines surrounded by /**xxx*/ where xxx are any characters contained in the string specified with this switch. This DGL switch has the same effect as the /O argument switch.
/P	During compilation, assume that the correct number of arguments will always be passed to all external procedures. This eliminates the need for many runtime checks.
/Q	Allow the use of question marks (?) in identifier names.
/R	Use floating-point arithmetic to perform all integer division within subexpressions. This increases accuracy by reducing rounding and truncating errors.
/REV=rev-num	Enter a revision number for an .OB or .RB object file. The default value is the current DG/L compiler revision number. For RDOS, AOS, or 16-bit AOS/VS, the format for rev-num is major-rev-num [.minor-rev-num]. For 32-bit AOS/VS, the format is major-rev-num *[.minor-rev-num [.update-num[.pass-num]]]*. For RDOS code, each number must be less than 100; for AOS and AOS/VS, each number must be less than 256.
/S	Generate code for full subscript checking. Note that an object program without subscript checking runs faster and requires less memory.
/T	Generate code for string overflow checking.
/TEMP=directory	Put temporary files in this directory. If the directory is on a fixed-head disk, compilation may be faster.
/V	Add information to the listing, giving the status of each line, i.e., the block level, whether the line is from an INCLUDE file, and whether the line compiled conditionally.
/W	Produce warning messages during compilation.
/WSAVS	(For MV/8000-32 only) Generate WSAVSs instead of WSAVRs.
/X	Generate a full cross-reference table, including constant references. Without this switch, only variables are cross-referenced.
/Y	Put constants into the shared code area, instead of the shared data segments of memory. This switch applies to AOS and AOS/VS-16 programs containing overlays.
/Z	Find all EXTERNAL integers in page zero of memory. This lets the compiler generate shorter object code.

DGL (continued)

Argument Switches

Note that each argument switch has an equivalent DGL command switch.

/B — Write the binary object code to the file specified by this argument. This argument switch has the same effect as the /O=pathname DGL command switch.

symbol/C — Generate code for the specified machine/operating system combination. You can append this argument switch to any of the codes given above. This argument switch has the same effect as the /CODE= DGL command switch.

/E — Write error messages to the file specified by this argument. This argument switch has the same effect as the /E=pathname DGL command switch.

/L — Write a listing to the file specified by this argument. This argument switch has the same effect as the /L=pathname DGL command switch.

string/O — Perform conditional compilation, using the string specified by this argument. This argument switch has the same effect as the /OPT=string DGL command switch.

Example

```
) XEQ DGL/G/L=TEST.LS/E=TEST.E/V TEST.DG )
```

The DGL command switch /G creates a local symbol block in the .OB file, used for debugging purposes. The /L switch sends the listing to file TEST.LS, and /E writes the error messages to TEST.E. The /V switch sends additional information about the lines of code to the listing file: the block level, whether the line is from an INCLUDE file, and whether it compiled conditionally.

DIRECTORY *Command*

Set or display the current directory setting.

Format

DIRECTORY *[pathname]*

Command Switches

/1= {IGNORE | WARNING | ERROR | ABORT} — Set CLASS1 to the specified severity level for this command.

/2= {IGNORE | WARNING | ERROR | ABORT} — Set CLASS2 to the specified severity level for this command.

/I — Set the working directory to the initial working directory.

/I pathname — Set the working directory to the directory specified by pathname, which starts from the initial working directory.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/P — Set the current working directory to the previous environment's directory (no arguments allowed).

/Q — Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) DIRECTORY ⤸
:UDD:USER
) DIRECTORY  BETA ⤸
) DIRECTORY ⤸
:UDD:USER:BETA
) DIRECTORY/I  GAMMA:DELTA ⤸
) DIRECTORY ⤸
:UDD:USER:GAMMA:DELTA
) DIRECTORY/I ⤸
) DIRECTORY ⤸
:UDD:USER
)
```

First, displays the working directory's pathname. Next, makes BETA the working directory and displays its pathname. Then, using the /I switch and a pathname, makes DELTA the working directory. Finally, uses the /I switch without an argument to set the working directory to the initial directory.

[!DIRECTORY] *Pseudo-Macro*

Expand to the current or previous environment's working directory.

Format

[!DIRECTORY *[pathname]*]

Macroname Switch

/I	Expand to the initial working directory setting.
/I pathname	Expand to the directory specified by pathname, which starts from the initial working directory setting.
/P	Use the previous environment's working directory.

Argument Switches

None

Example

```
) WRITE THE WORKING DIRECTORY IS & ⤸
&) [!DIRECTORY]. ⤸
THE WORKING DIRECTORY IS :UDD:DAN:
ALPHA.
)  WRITE [!DIRECTORY/I] ⤸
:UDD:DAN
) WRITE [!DIRECTORY/I TEST] ⤸
:UDD:DAN:TEST
)
```

DISCONNECT *Command*

Break a Customer-Server connection.

Format

DISCONNECT process-ID

The process ID must be the PID of a server process to whom you have previously been connected.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) DISCONNECT 12 ↵
)
```

Disconnects the user from the server process whose ID is 12.

DISMOUNT *Command*

Ask the system operator to physically dismount a tape.

Format

DISMOUNT linkname *[message]*

The DISMOUNT command requests the operator to dismount a tape. Use it when you're finished using a tape that you earlier asked the operator to mount with the MOUNT command.

For linkname, use the name you chose as a linkname when you typed the MOUNT command. (If you specify a different name, the CLI will display a *FILE DOES NOT EXIST* message. If you can't remember the name, try typing F/AS/TYPE=LNK :UDD:[!USERNAME] ↵.)

The DISMOUNT command deletes the link created by the MOUNT command. The system rewinds the tape (if needed) and restores the tape unit ACL to what it was before the tape was mounted.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) MOUNT MYTAPE PLEASE MOUNT TAPE XB43 ⤸
(Person at system console mounts tape and notifies EXEC.)
) LOAD/N MYTAPE:0 ⤸
(The system displays filenames in file 0 of the tape.)
) LOAD/V MYTAPE:0 OLD_ADDRESSES ⤸
(pause)
OLD_ADDRESSES
) DISMOUNT MYTAPE THANKS -- PLEASE FILE& ⤸
&)□THE TAPE. ⤸
)
```

This example shows a user requesting a tape mount, loading a file from the tape, and requesting a tape dismount. It shows access to unlabled tape. For an example with labeled tape, see the MOUNT command.

DISPLAY *Utility*

Print a file in octal and ASCII values.

Format

```
XEQ DISPLAY input-pathname
    [destination-pathname]
```

This utility produces a copy and/or a listing of the input file.

The default listing format is 16 input characters per line, printed first as octal values and then as text. The ASCII characters that the CLI cannot print as text (those whose octal value is less than 40 and greater than 176_8) print as a period (.). Repeated identical lines print as ****

The default values for the input parameters are FIRST=0, INCREMENT=1, and LAST=32767. If the input is on tape, the tape input block size is the size specified for the device when the system was generated.

If you specify a tape destination pathname, the default values for the output parameters are: OBLOCKSIZE= the block size of the input tape; and ODENSITY=0. DISPLAY will create the destination file if it does not already exist. If it does exist, DISPLAY will delete the file and then recreate it.

DISPLAY Switches

/ALL — Process all files to logical end-of-tape on the input tape. Use this switch only for input files on tape. The destination file, if specified, may be either another tape or a disk file. If the destination file is a tape, each file on the input tape will be copied to a separate file on the output tape. If the destination is a disk file, all files except the first one will be appended to the disk file. The first file will either replace the destination file (default), or be appended to the file (/APPEND switch). The tapes must be specified without a file (e.g., XEQ DISPLAY @MTB0 @MTC10 will copy all the files from @MTB0 to @MTC10).

/APPEND — Append the output to the destination file if it exists. The default is to first delete the destination file and then recreate it.

DISPLAY (continued)

/BYTE — List the file in byte format (octal values) with no text.

/CONVERT — Convert EBCDIC input into ASCII.

/DECIMAL — List the file in decimal values rather than octal values.

/FIRST=m — First block to be processed, where m is greater than or equal to 0 and less than or equal to 32767.

/HEXADECIMAL — List the file in hexadecimal values rather than octal values.

/IGNORE — Ignore the logical end-of-tape on the input file.

/INCREMENT=i — Process every i^{th} block in the input file.

/IPHYSICALEOT — Ignore physical end-of-tape (EOT) on the input file and go to the next EOF.

/L — Append output to current list file instead of to @OUTPUT.

/L=pathname — Append output to the file specified by pathname instead of to @OUTPUT.

/LAST=n — Last block to be processed, where n is greater than m and less than or equal to 32767.

/LISTUDA — List the UDA of the input file.

/NOLIST — Suppress the listing file.

/NUMERICONLY — List the numeric value of the input file only, and not the text value.

/OBLOCKSIZE=b — Specify the block size of the destination tape, where b is the number of bytes.

/ODENSITY=d — Specify the density of the destination tape. Use this switch only with variable-density units (like MTBs). The following are valid values for d:

Value	Density
800	800 BPI
1600	1600 BPI
6250	6250 BPI (AOS/VS only)
ADM	Automatic Density Matching

/TEXTONLY — List the text value of the input file only, and not the numeric value.

/UPPERCASE — Convert any lowercase characters to uppercase in the text part of the listing.

/WIDTH=j — Set the width of the lines in listing file, where j represents the number of characters per line. j must be an even number and less than or equal to 124.

Example

```
) XEQ DISPLAY/L=@LPT DATA76 ↵
```

Produces an octal value and ASCII listing of the file DATA76 on the line printer.

```
) XEQ DISPLAY/ALL/HEXADECIMAL @MTB0 ↵
```

Produces a hexadecimal and ASCII listing of all the files on MTB0.

```
) XEQ DISPLAY/CONVERT/NOLIST @MTB0 DFILE ↵
```

Creates an ASCII file on disk of the first EBCDIC file on the tape.

DUMP *Command*

Dump one or more files from the working directory to the specified dump file.

Format

DUMP dumpfile *[source-pathname]...*

You may use templates for the *source-pathname* argument(s). If you supply no *source-pathnames*, the template # is assumed. Unless you use the /FLAT switch, the directory structure of the dumped files will be maintained.

Command Switches

/1= (IGNORE | WARNING | ERROR | ABORT) — Set CLASS1 to the specified severity level for this command.

/2= (IGNORE | WARNING | ERROR | ABORT) — Set CLASS2 to the specified severity level for this command.

/BUFFERSIZE=bytes — Blocks dumped to the tape will have length of **bytes**.

/DENSITY=mode — Control the magnetic tape density of your dump tape. Use this command with variable-density units, like (MTBs). The following are the **mode** options:

Mode	Density
800	800 BPI
1600	1600 BPI
6250	6250 BPI (AOS/VS only)
ADM	Automatic Density Matching

/FLAT — Do not maintain tree structure; dump all files from the specified directories as one directory.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=@LPT — List file and directory names of files dumped to the line printer.

/L=pathname — Write CLI output to the file specified by **pathname** instead of to @OUTPUT.

/NACL — Dump files without ACLs (later when you LOAD the files, they will be given default ACLs).

/Q — Set SQUEEZE to ON for this command.

/RETAIN=days — Set the retention period of a labeled tape to **days**. This switch applies only to labeled tape, and labeled floppy diskettes.

/SPECIFIC=volid — Request a specific tape volume to be dumped to. EXEC will display a message on the system console that requests the tape volid be mounted. Normally in a multivolume labeled tape dump, the system dumps to tape volids in the sequence you specify with the MOUNT/VOLID= command. This switch allows you to select a specific tape volume to receive a labeled tape dump.

/TYPE=type — Select all files of the specified type. Types are provided in Table 2-6. **type** can be in the following forms:

XXX	3-letter mnemonic
n	decimal number (0, 10-13, or 64-255)
m-n	decimal numbers that define a range of file types
\n	decimal number to exclude
\m-n	decimal numbers that define a range of file types to exclude

DUMP (continued)

	You can use more than one /TYPE= switch in a command line.
/V	Verify dumped files on @OUTPUT.

The following switches tell the CLI to dump all files that meet the specified conditions.

/AFTER/TLA=	Dump only the files accessed after the specified time, date, or date:time. date:time is in the form dd-mm-yy:hh:mm:ss.
/AFTER/TLM=	Dump only the files modified after the specified time, date, or date:time. See /AFTER/TLA= for format.
/BEFORE/TLA=	Dump only the files accessed before the specified time, date, or date:time. See /AFTER/TLA= for format.
/BEFORE/TLM=	Dump only the files modified before the specified time, date, or date:time. See /AFTER/TLA= for format.
/BEFORE/TLM=date	Dump only the files last modified before the specified date. Date is in the form dd-mmm-yy.
/BEFORE/TLM =date:time	Dump only the files modified before the specified time and date. See AFTER/TLA= for format.
/BEFORE/TLM=time	Dump only the files modified today before time. Time is in the form hh:mm:ss.

NOTE: You may specify a range of dates by using both the /BEFORE and /AFTER switches. However, you must use the same modifier for both: either /TLM= or /TLA=. You cannot use /TLM= and /TLA= at the same time.

Argument Switches

None

Example

```
) DUMP/V/FLAT/NACL  FILE6.DUMP  :UTIL:+.PR ↵
19-JUN-84   10:19:50
DISPLAY.PR
SPEED.PR
EXEC.PR
XLPT.PR
LINK.PR
SED.PR
)
```

Dumps all program files in the utilities directory, without their ACLs, to disk file FILE6.DUMP; verifies dumped files.

```
) DUMP/AFTER/TLM=4-JUL-84:11:03:42  @MTB0:3 ↵
)
```

Dumps all files in the working directory that were last modified after 11:03:42 a.m. on July 4, 1984 to file 3 of the tape mounted on @MTB0.

```
) DUMP/V/SPECIFIC @LMT:VOL4:FILE0  +.SR ↵
)
```

This command dumps all files with the .SR extension onto labeled tape starting with volid VOL4. The /SPECIFIC switch indicates that VOL4 is not the first volid in the set.

DUMP_II *Utility*

Dump one or more files from the working directory to the specified dump file (AOS/VS only).

Format

XEQ DUMP_II dumpfile *[source-pathname] ...*

The DUMP_II utility does just what the DUMP command does — but faster. The dumpfile is specified in the same manner as in the DUMP command. DUMP_II accepts filename templates for the files to dump exactly as DUMP does. All of the CLI DUMP command's switches are supported.

The only difference between DUMP_II and DUMP is that you type XEQ DUMP_II instead of DUMP. (File DUMP_II.PR is supplied in directory :UTIL.)

For example, for DUMP you might type

```
) DUMP/V/L=FILESDUMPED @MTB0 & )
&) UDD:MYDIR:# )
```

and for DUMP_II you would type

```
) XEQ DUMP_II/V/L=FILESDUMPED & )
&) @MTB0 UDD:MYDIR:# )
```

Either of these tapes could then be loaded back with the CLI LOAD command.

DUMP_II is faster than DUMP; it also uses more system resources than DUMP. The amount of time DUMP_II saves varies, depending on the size of the files being dumped, the type of tape unit, the amount of memory DUMP_II is allowed to use, and other things. Typically — on an MTB tape unit — DUMP_II takes 50–60% of the time that DUMP takes.

DUMP_II Switches

Switches are identical to DUMP's, with the following additions:

/ERROR=pathname — Send error messages to pathname. Also, messages will be written to the console or batch output file, and, if specified the listing file.

/MEMORY=value — Control the amount of memory that DUMP_II can use for its data buffers. The numeric value range is 0 to 200; the default is 100. Valid values are

0 to 200	Integer numeric value
MIN	Smallest amount of memory; slowest dump
LOW	Recommended value for a model 6125 (MTC) tape drive
MED	Provides good all around performance
HIGH	Provides better performance than MED; with great overhead
MAX	Provides best performance (with the greatest overhead) if used with a large memory configuration

/STATISTIC — Print summary statistics on the console, or send to the listing file if you include the /L switch. The format varies with other switches (see Examples).

Argument Switches

None

Example

```
) X DUMP_II/BUFFER=8192/DENSITY=1600 & )
&) /V/L=@LPB @MTB0:0 )
```

This dumps all files in and beneath the working directory to file 0 (the first file) of the tape on unit MTB0. The buffer size of 8192 uses less tape than the default size (but if the tape file is later restored to disk with LOAD, the same buffer size must be specified). The density of 1600 bpi ensures that the dump will occur at 1600 bpi (otherwise, if the unit density switch were set at low, the dump might occur at 800 bpi). The /V/L=@LPB sends a listing of all files dumped to the line printer.

DUMP_II (continued)

```
) QBATCH X DUMP_II/MEMORY=LOW & )
&) /STATISTICS/V/L=XX #:+FILE+ & )
&) @MTC1:3 )
```

This dumps all files that match the template +FILE+ in and beneath the working directory, to tape file 3 on unit MTC1. The dump listing and statistics go to file XX in the working directory.

The statistics listing might look like this:

Statistics

Number of files dumped	-	*65*	
Total bytes written to tape	-	*1197617*	
Number of bytes used for file data	-	*1191129*	*99.4%*
Number of bytes used for file information	-	*4310*	*0.3%*
Number of bytes used for internal logic	-	*2178*	*0.1%*
Total data bytes read	-	*1263833*	
Number of data bytes written	-	*1191129*	*94.2%*
Number of zero bytes compressed	-	*72704*	*5.7%*

The number of bytes used for file data represents the contents of disk files. Bytes used for file information were used for file information, like filename, access control list, date created, and so on. Bytes used for internal logic means bytes of control information written to the tape to form the dump format.

[!EDIRECTORY] *Pseudo-Macro*

Expand to the directory portion of a pathname.

Format

[!EDIRECTORY pathname *[pathname]...*]

This pseudo-macro expands to the directory portion of a pathname. By *directory portion,* we mean the pathname string from the left-most character up to, but excluding, the right-most colon or prefix character.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!EDIRECTORY :UDD:AOS:MODS:& )
&) SCALL.SR] )
:UDD:AOS:MODS
) WRITE [!EDIRECTORY @CON32 =MODS:FILE & )
&) :CLI.PR] )
@ =MODS :
)
```

The first example shows the directory portion of the full pathname :UDD:AOS:MODS:SCALL.SR. The second example shows the directory portions of the three pathnames used as input to the !EDIRECTORY pseudo-macro.

[!EEXTENSION] *Pseudo-Macro*

Expand to the extension portion of a pathname.

Format

[!EEXTENSION pathname *[pathname]...*]

This pseudo-macro expands to the extension portion, also known as suffix portion, of a pathname. By *extension portion,* we mean the input string from the last period after the right-most colon or prefix character to the end of the string, inclusively. If there is no period in the string, this pseudo-macro returns a null.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!EEXTENSION :UDD:AOS:MODS:SCALL.SR] ↓
.SR
) WRITE [!EEXTENSION :UDD:CLI:SRC.SR.BU] ↓
.BU
) WRITE [!EEXTENSION @CON32 & ↓
&) =MODS:FILE :CLI.PR] ↓
.PR
)
```

The first example shows the extension .SR. The second example shows .BU, the characters that follow the last period after the right-most colon. The third example shows three pathnames. Since only the last pathname has an extension, the CLI returns the proper extension for that pathname, and null for the first two pathnames.

[!EFILENAME] *Pseudo-Macro*

Expand to the filename portion of a pathname.

Format

[!EFILENAME pathname *[pathname]...*]

This pseudo-macro expands to the filename portion of a pathname. By *filename portion,* we mean the input string from the right-most colon or prefix character to the end of the string. The filename includes all extensions.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!EFILENAME :UDD:AOS:MODS:SCALL.SR] ↓
SCALL.SR
) WRITE [!EFILENAME :UDD:CLI:SRC.SR.BU] ↓
SRC.SR.BU
)
```

[!ELSE] *Pseudo-Macro*

Include CLI input conditionally.

Format

[!ELSE]

This pseudo-macro does not accept arguments. You can use [!ELSE] only after one of the pseudo-macros that begin conditional branching.

If the condition stated in the initial pseudo-macro is true, then the CLI executes the input lines that appear before the !ELSE pseudo-macro and does not execute the input lines that appear after !ELSE. If the initial condition is false, then the CLI skips the input between the initial pseudo-macro and !ELSE, and executes the input lines which appear after !ELSE up to the next !END pseudo-macro.

Macroname Switches

None

Argument Switches

None

Example

```
) [!EQUAL,1,2]WRITE EQUAL & ↓
&) [!ELSE]WRITE NOT EQUAL & ↓
&) [!END] ↓
NOT EQUAL
)
```

Since the !EQUAL pseudo-macro is false, the CLI executes the command(s) that follow(s) the !ELSE pseudo-macro.

See Chapter 5 and *Two Examples Using CLI Environment Levels* in Chapter 4 for additional examples of the !ELSE pseudo-macro.

[!ENAME] *Pseudo-Macro*

Expand to the name portion of a pathname.

Format

[!ENAME pathname *[pathname]...*]

This pseudo-macro expands to the name portion of a pathname. By *name portion,* we mean the input string after the right-most prefix (: @ = ^) up to, but excluding, the right-most period. If there are any filename extensions, this pseudo-macro expands to all but the very last one. If there are no prefix characters in the string, the name begins with the left-most character in the string. If there is no period in the string, the name ends at the right-most character.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!ENAME :UDD:APS:MODS:SCALL.SR] ↓
SCALL
) WRITE [!ENAME :UDD:CLI:SPR.SR.BU] ↓
SPR.SR
) WRITE [!ENAME @CON32 =MODS:FILE :CLI.PR] ↓
CON32 FILE CLI
)
```

[!END] *Pseudo-Macro*

End an !EQUAL or !NEQUAL macro loop.

Format

[!END]

This pseudo-macro does not accept arguments. Use !END to terminate the sequence of CLI input that follows one of the conditional branching pseudo-macros. See Chapter 5 for a discussion of conditional pseudo-macros.

Macroname Switches

None

Argument Switches

None

Examples

```
) [!EQUAL,1,1]WRITE EQUAL[!END] )
EQUAL
) [!EQUAL,1,2]WRITE EQUAL[!END] )
)
```

The CLI executes the first command line because the condition is true; it does not execute the second because the condition is false.

Notice that there are no spaces between the [!EQUAL] or [!END] statement and the following command, WRITE in this case.

ENQUEUE *Command*

Queue one or more file entries to a spoolable output device (if your system has no EXEC process).

Format

ENQUEUE device-pathname *[pathname]...*

The operator must have enabled spooling to the device you want to use. The CLI places an entry for each file in the spooler output queue for that device.

You may use templates in the pathname arguments.

Command Switches

/1=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS1 to the specified severity level for this command.
/2=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/V		Display the names of the queued files.

Argument Switches

/B	Output in binary mode; do not interpret special control characters (such as TAB); default is output in text mode.
/D	Delete disk file after output; by default, the system retains the original file.
/H	Output header; default suppresses header at the top of each page.
/MES=x	Output message x to operator console; default is no message.
/P	Pause for operator response before outputting file; default is immediate output.

ENQUEUE (continued)

If you specify /P (and no /MES=x), the spooler process displays

PAUSE FROM queuename

on the operator terminal before output.

If you specify /MES=x (and omit /P), the spooler process displays the message

FROM queuename:

on the operator terminal and then outputs the file.

If you use both /MES=x and /P, then, before it starts the output, it prefixes the message displayed on the operator terminal with

PAUSE FROM queuename.

The operator must respond with an appropriate control command, such as

```
CONTROL @SPOOL CONTINUE
```

to start file output.

Example

```
) ENQUEUE @LPT OUTPUT.LS/P/MES=TWO_ & ↓
&) PART_FORMS ↓
)
```

Queue OUTPUT.LS to the line printer. Before the line printer begins, the system pauses and displays the message *TWO_PART_FORMS.*

[!EPREFIX] *Pseudo-Macro*

Expand to the prefix portion of a pathname.

Format

```
[!EPREFIX pathname /pathname.../]
```

This pseudo-macro expands to the prefix portion of a pathname. By *prefix portion,* we mean the input string from the left-most character up to, and including, the right-most prefix character (: @ = ^). If there is no prefix character in the string, the pseudo-macro returns a null.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!EPREFIX :UDD:AOS:MODS:SCALL.SR] ↓
:UDD:AOS:MODS:
) WRITE [!EPREFIX @CON32 =MODS:FILE :CLI.PR] ↓
@ =MODS: :
) WRITE [!EPREFIX MYFILE] ↓
)
```

[!EQUAL] *Pseudo-Macro*

Include input conditionally.

Format

```
[!EQUAL argument1, argument2]
```

This pseudo-macro begins a sequence of text that the CLI is to conditionally execute. You must end the sequence with the !END pseudo-macro. The sequence can optionally include the !ELSE pseudo-macro.

The !EQUAL pseudo-macro must always have two arguments. !EQUAL compares the two arguments character by character. If they match, the CLI executes the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI doesn't execute the input following it up to the !END pseudo-macro.

If the arguments don't match, the CLI does not execute the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI executes the input following the !ELSE up to the !END.

See Chapter 5 for a discussion of conditional pseudo-macros.

Macroname Switches

None

Argument Switches

None

Example

Given a macro containing

```
[!EQUAL,%1%,*]
WRITE STAR
[!ELSE]
WRITE NOT STAR
[!END]
```

This will write *STAR* if you call the macro with the argument *; otherwise, it will write *NOT STAR*.

Note that you can also code the macro as follows

```
WRITE [!EQUAL,%1%,*]STAR[!ELSE]NOT STAR[!END]
```

Notice that we used commas to separate the arguments in the !EQUAL pseudo-macro. If we used spaces, and argument 1 was null (or not present), the spaces on either side of the %1% would have become a single delimiter, giving [!EQUAL,*]. This format is invalid, since the !EQUAL pseudo-macro takes exactly two arguments. Notice that there are no spaces between the bracketed !EQUAL statement and other commands and arguments.

EXECUTE *Command*

Execute a program.

Format

EXECUTE pathname *[argument-to-new-program]...*

The CLI creates a subordinate swappable process with the same priority and privileges that it has. It takes the subordinate process's program from the program file that you specify in the command line. The arguments to the new program are placed in the initial IPC message to the new process. The program can refer to these arguments by using the ?GTMES system call (see Appendix B for details). The subordinate process's generic @INPUT, @OUTPUT, and @CONSOLE are the same as its parent's (the CLI's). The subordinate process's generic @LIST and @DATA files are the current list file and data file settings. (Note that these are not necessarily the same as the CLI's generic @LIST and @DATA files.)

The CLI first tries to execute pathname.PR. If that fails, the CLI tries pathname.

The CLI is blocked until the subordinate process terminates. The CLI may take exceptional action depending on whether or not the subordinate process returns exceptional condition flags (see Appendix A).

Command Switches

/1= {IGNORE / WARNING / ERROR / ABORT} — Set CLASS1 to the specified severity level for this command.

/2= {IGNORE / WARNING / ERROR / ABORT} — Set CLASS2 to the specified severity level for this command.

/I — Create input for pathname from @INPUT. The last line must contain a single) .

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/M — Create input for pathname from the macro body. The last line of the macro body must contain a single) .

/Q — Set SQUEEZE to ON for this command.

/S — Store the program termination IPC message in the current STRING (instead of displaying it to @OUTPUT).

Argument Switches

Use any argument switches appropriate for the program specified in pathname.

Example

```
) EXECUTE MYSORT )
```

Runs a program named MYSORT.

```
) EXECUTE/S PROG1 )
```

Runs a program named PROG1 and places its termination message in STRING.

[!EXPLODE] *Pseudo-Macro*

Expands arguments into single-character arguments.

Format

[!EXPLODE argument *[argument]...*]

This pseudo-macro interprets its arguments as a single string; it converts all spaces and tabs into commas (the CLI delimiter) in accordance with standard CLI rules. !EXPLODE expands the string and inserts commas between every pair of characters (including commas) in the original string. Use !EXPLODE to access characters of arguments as individual arguments.

Macroname Switches

None

Argument Switches

None

Example

[!EXPLODE ABC]

Expands to A, B, C

[!EXPLODE A]

Expands to A

[!EXPLODE A,C]

Expands to A,,,C

[!EXPLODE[!TIME]]

Expands to 0,9,:,2,3,:,4,7

[!EXPLODE[!DATE]]

Expands to 2,6,-,N,O,V,-,8,0

In a macro

WRITE [!EXPLODE ABC]

The system responds with

A B C

F5 *Language*

Compile a FORTRAN 5 source file.

Format

XEQ F5 source-pathname

or

F5 source-pathname

You can use either the XEQ F5 command line or the F5.CLI macro provided to you to compile a FORTRAN 5 program.

The F5 macro is used to compile a FORTRAN 5 source file. The macro first searches for source-pathname.FR. If that is not found, it searches for source-pathname.

Each FORTRAN 5 main program, subroutine, and subprogram must be compiled separately. Output will be an object file (binary output) named pathname.OB. By default, compilation produces no listing.

You link the separate FORTRAN 5 modules into an executable program by using the F5LD macro, which invokes the Link utility.

Under AOS/VS, FORTRAN 5 runs as a 16-bit process. In addition, F5LD is a link to the F5LDVS16.CLI macro.

For a complete description of the FORTRAN 5 programming language and the CLI F5 command line, see the *FORTRAN 5 Reference Manual* (093-000085) and the *FORTRAN 5 Programmer's Guide (AOS)* (093-000154).

F5 Switches

/B	Produce a brief listing: the input source program, the storage map, the list of subprograms called, the cross-reference, and the error list. The generated code is not included.
/C	Check the syntax of the source program. The source program and error list are sent to the listing file, if one is specified. The error list is also sent to the error file, if one is specified.

F5 (continued)

/D or /DEBUG	Debug. Compile code that allows the long form error traceback or routine to output line numbers. Do not use this switch when compiling the final version of your program.
/E[=*pathname*] or /ERRORS [=*pathname*]	Write errors to *pathname*. If you use /E without specifying a pathname, error messages are suppressed. If you do not use this switch, error messages are written to @OUTPUT.
/I	Do not list source lines from INCLUDE files.
/L	Write the listing to the current list file.
/L=pathname	Write the listing to the file specified by pathname.
/N	Do not produce an object file.
/NOLEF	Do not generate Load Effective Address instructions (LEFs). This switch is useful if you are using I/O instructions in assembly language routines combined with FORTRAN 5 programs.
/O or /OBJECT=pathname	Name the object file pathname.
/P	Use punched card format. The first 72 characters of each input line are used as FORTRAN 5 source code, although the entire line is sent to the listing file.
/S or /SUBCHECK	Generate code to check subscript references. A runtime or routine determines if a reference is outside the range of an array.
/X	Compile lines with an X in column 1. If you do not use this switch, the compiler treats these lines as comments.

Argument Switches

None

Example

```
) F5 MYPROG ↵
```

Compiles either MYPROG.FR or MYPROG, depending on whether the source filename has the .FR extension. The compile produces the object file MYPROG.OB.

```
) F5/E/I/L=PROG.LS PROG ↵
```

Compiles either PROG or PROG.FR. Generates a listing file, PROG.LS, and does not include lines from INCLUDE files in the source listing. Suppresses all error messages.

```
) F5 MAIN ↵
) F5 SUB ↵
) F5 XFUN ↵
) F5 XSUB ↵
) F5LD MAIN SUB XFUN XSUB ↵
```

Compiles separately the four modules that make up the complete program. Then uses the macro F5LD.CLI to invoke the Bind utility, which builds the executable program. (This example is valid only for AOS. For an AOS/VS example, replace the last line with a line that invokes the Link utility.)

F5LD *Language*

Link object modules to form an executable FORTRAN 5 program.

Format

F5LD objectmodule *[objectmodule]...*

F5LD is a macro that invokes the Link utility to make FORTRAN 5 object modules into an executable program. (Under AOS/VS, F5LD is actually a link to the macro F5LDVS16.CLI.)

In general, the object modules appear on the command line in the following sequence:

- Main FORTRAN 5 program
- User subprograms and optional user modules
- Support libraries (e.g., Commercial Subroutine Package)

In addition, if you use QCALLS within the FORTRAN 5 modules, you must add the FORTRAN 5 library F5ASYS.LB to the F5LD command line.

For a complete description of the FORTRAN 5 programming language and the F5LD command line, see the *FORTRAN 5 Reference Manual* (093-000085) and the *FORTRAN 5 Programmer's Guide* (093-000154).

F5LD Switches

/B	Produce a listing of the symbol file, with symbols ordered both alphabetically and numerically.
/E	List all numbers in hexadecimal.
/L	Write a listing to the current list file.
/L=pathname	Write a listing to the file specified by pathname.
/P=pathname	Name the executable program file pathname.PR. By default, the file assumes the name of the first object module in the command line, plus the .PR extension.

Argument Switches

/S Convert this shared code module to an unshared code module.

/U Bind local symbols from this module into the symbol file. /U works only if you used it for this module in an earlier macroassembler command. The FORTRAN 5 compiler outputs no local symbols.

Example

```
) F5LD/L=NEWPROG.LM/P=NEWPROG.PR &
&) MYPROG
```

Builds compiled FORTRAN 5 module MYPROG.OB into an executable program named NEWPROG.PR. In addition, writes a listing to the file named NEWPROG.LM.

F77 *Language*

Compile a FORTRAN 77 source file.

Format

For AOS:

```
F77 source-pathname
```

or

```
F77 (source-pathname [source-pathname]... )
```

For AOS/VS:

```
F77 source-pathname [source-pathname]...
```

The F77 macro is used to compile a FORTRAN 77 source file. The compiler looks first for a file named pathname.F77, and if not found, for a file named pathname.

The F77 macro supplied with AOS includes the /INTEGER=2 and the /LOGICAL=2 switches. You can override them by explicitly appending the switch to the command and specifying the value 4 (for example, F77/INTEGER=4 MY_PROG). If you append one of these switches to the F77 macro, you must use the entire switch name, not an abbreviation. (You can use unique abbreviations for all other AOS switches, and for all AOS/VS switches.) The macro supplied with AOS/VS does not include the /INTEGER= or /LOGICAL= switches.

For a complete description of the FORTRAN 77 programming language and its compiler command, see the *FORTRAN 77 Reference Manual* (093-000162).

F77 Switches

/CARDFORMAT	Impose card format. Read only the first 72 characters of each line, and blank pad lines with fewer than 72 characters to 72 characters. Treat all characters read as significant. You may place any other text, such as sequence numbers, in columns 73–80.
/CODE	Generate an assembly language listing of the program and write it to the file specified by the /L switch. If you omit /L, the /CODE switch is ignored.
/DEBUG	Generate symbols and code for later use by the SWAT high-level debugger. Do not use this switch when compiling the final version of a program.
/DOTRIP=1 or 0	If DOTRIP=1, force each DO loop to execute at least once. By default, the compiler generates DO loops that will not execute if they do not need to execute. Some user programs, written for early ANSI standards, expect the results produced by specifying /DOTRIP=1.
/E=pathname	Write compilation errors to the file specified by pathname and to any listing file, instead of to @OUTPUT.
/ERRORCOUNT=n	Report and count both warning and error messages. When this count exceeds the value of n, the compiler terminates. n is a positive integer between 1 and 32767. The default value is /ERRORCOUNT=100.
/HOLLERITH=ANSI or NON_DG or OLD_DG	Affect the storage of character and Hollerith constants both in expressions and as arguments to subprograms. The possible values for the /HOLLERITH switch are ANSI NON_DG OLD_DG ANSI is the default value.
/INTEGER=2 or 4	Set the default integer length for this compilation to the specified number of bytes. Macro F77.CLI supplied with AOS contains the switch /INTEGER=2, but you can explicitly specify /INTEGER=4.
/L	Generate a program listing and write it to the current list file.
/L=pathname	Generate a program listing and write it to the file specified by pathname.

/LINEID	(AOS/VS only) Generate code that will identify a line causing a runtime error. /LINEID includes the function of the /PROCID switch. Do not use this switch when compiling the final version of a program.
/LOGICAL=2 or 4	Set the default logical length for this compilation to the specified number of bytes. Macro F77.CLI supplied with AOS contains the switch /LOGICAL=2, but you can explicitly specify /LOGICAL=4.
/N	Scan the source file as usual, but do not produce an object file. This switch is useful for a quick syntax check.
/NOFNS	Suppress the generation of FNS (floating no-operation) instructions following FFAS instructions.
/NOLEF	Instruct the compiler to avoid the use of LEF instructions in the code generated for each program unit compiled. Use this switch if you intend to disable LEF mode from within your program (AOS, F7716, MP/AOS, and MP/AOS-SU only).
/NOMAP	(AOS/VS only) Do not include a storage map of program entities as part of the listing file.
/NOWARNINGS	Suppress the reporting and counting of warning messages. Use this switch with caution because changing a program in one place may unexpectedly cause problems in another place.
/O=name	Use this switch if you want the .OB file to have a name different from the default name of source-filename.OB.
/OPTIMIZE *[=1 or 2 or 3]*	Set optimization to the specified level, ranging from 1 (lowest) to 3 (highest). If you omit the switch, the compiler performs no optimizations. If you include the switch but do not specify a level, the default level is 3. Do not use this switch along with /DEBUG, /PROCID, or /SUB.
/PROCID	(AOS/VS only) Include all procedure names in the object file. At runtime, FORTRAN 77 will be able to report the program unit in which an error occurs. /LINEID includes the function of the /PROCID switch.
/SAVEVARS	Generate code to save all entities on returns from subprogram units, and generate all storage as static. This switch has the effect of a SAVE statement in each unit. Named and blank COMMON blocks, values passed in dummy arguments, and values assigned with DATA are always saved, even if you do not use this switch.
/STATISTICS	Write statistics (such as number of lines compiled) to @OUTPUT. If you specify /L, the statistics are also included in the listing file.
/STRINGS=ANSI or DG	If /STRINGS=ANSI, accept angle brackets and the characters within them as literals. By default, the compiler interprets angle brackets as control characters. This switch, with the ANSI value, lets you use angle brackets as literal printed characters so that <7> results in 3 displayed characters on a console instead of a beep sound.

F77 (continued)

/SUB	Generate code for subscript and character substring checking. At runtime, the program reports any subscript or substring references that are out of range. Do not use this switch when compiling the final version of a program.
/XREF	(AOS/VS only) Generate an alphabetical listing of all program entities and include it in the listing. If you omit /L, the /XREF switch is ignored.

Argument Switches

None

Example

) F77 MYPROG ↵

Compiles either MYPROG.F77 or MYPROG, depending on whether the source filename has the .F77 extension. The compiler will display all errors on @OUTPUT (the console), will not produce a listing file, and will generate object file MYPROG.OB.

) F77/SUB/XREF/L=@LPT PROG ↵

Compiles either PROG or PROG.F77. Generates code for subscript and substring checking. Generates a listing file with alphabetical cross references, and sends the listing to the line printer.

) F77/OPTIMIZE PROG ↵

Compiles either PROG or PROG.F77. By default, optimization is set to 3, the highest level.

F77LINK *Language*

Link object modules to form an executable FORTRAN 77 program.

Format

F77LINK objectmodule *[objectmodule]...*

F77LINK is a macro that invokes the Link utility to make FORTRAN 77 object modules into an executable program. Use this macro after you have compiled a FORTRAN 77 program with the F77 macro.

You can not abbreviate either the F77LINK macroname or it's switches.

You can use any LINK switch as an F77LINK switch. Switches pertaining to F77LINK are given below. F77LINK passes all switches down to LINK.

For a complete description of the FORTRAN 77 programming language and it's LINK commands, see the *FORTRAN 77 Reference Manual* (093-000162).

F77LINK Switches

/AOS*[=revision_number]*	For the cross-compiler only (F7716).
/CHANNELS=n	Increase the value of n, if your program has more than 8 units open (including preconnected ones). For AOS, F7716, MP/AOS, and MP/AOS-SU only, this switch has a default value of 8.
/CIS	(AOS only) Use the commercial instruction set (CMP and FINT). This allows faster character comparisons and truncation of real numbers. Use this switch only if you have the commercial instruction set.
/DEBUG	(AOS/VS only) Include code for the SWAT high-level debugger. For SWAT to work, you must have compiled using the /DEBUG switch.

/FINT	(AOS and F7716 only). Direct Link to include modules from the F77 runtime library which will directly execute the FINT instruction. By default, F77 simulates FINT in software.
/FPF	(AOS and F7716 only.) Direct Link to include, from the F77 runtime libraries, modules that will use the Floating-Point Function instructions in the M600 instruction set. Use this switch only if you will run the program on an M600.
/IOCONFLICT	You may need this switch for multitasked programs. You may not use it under MP/AOS and MP/AOS-SU. See your Environment Manual for its explanation.
/PRECONNECTIONS=*	Use only the preconnections to units 5 and 6.
/PRECONNECTIONS =NONE	Sever all preconnections.
/PRECONNECTIONS =pathname	Use the user-defined preconnections in the .OB file specified by pathname.
/ROUND	(AOS/VS only) Direct the ECLIPSE MV/Family floating-point unit to use rounding for floating-point values. /ROUND is the default; /TRUNCATE is an option.
/TASKS=n	You need this switch for multitasked programs. See your *Environment Manual* for its explanation.
/TRUNCATE	(AOS/VS only) Direct the ECLIPSE MV/8000® floating-point unit to truncate floating-point values.

Argument Switches

None

Example

```
) F77LINK MYPROG )
```

Builds compiled FORTRAN 77 module MYPROG.OB into an executable program, MYPROG.PR.

```
) F77LINK/CHANNELS=12/DEBUG MYPROG1 & )
&) SUBA SUBB MYLIB1.LB )
```

Builds executable program file MYPROG1.PR from main program MYPROG1.OB, subroutines SUBA.OB, SUBB.OB, zero or more subroutines in library file MYLIB1.LB, FORTRAN77 library files and the common language runtime library. At most 12 units may be open at once, and the program executes under the SWAT debugger.

FCU *Utility*

Set nonstandard forms parameters for files to be printed.

Format

XEQ FCU

The FCU (Forms Control Utility) allows you to specify horizontal tabs and vertical forms settings for a user file or a forms entry in directory :UTIL:FORMS. You can create a forms entry in any directory, but the file must reside in :UTIL:FORMS for EXEC to use it. You can run the FCU whenever you wish to create, edit, or list forms control specifications.

Note that, to create forms control specifications, you must have created a file before invoking the FCU; the FCU does not itself create the file. The specifications are part of the file's UDA. Therefore, if the file size was 0 before you invoked the FCU, it will still be 0 after you have created the specifications.

XEQ FCU

This calls the Forms Control Utility. The FCU displays something like the following message on your terminal:

AOS (or AOS/VS) FORMS CONTROL UTILITY REVISION xx.xx date time

TYPE 'HELP' FOR INSTRUCTIONS.

COMMAND?

You can now type any of the following FCU commands followed by a NEW LINE:

Command Mnemonic	Meaning
B	Terminate the Forms Control Utility
C	Create forms control specifications for an existing file
E	Edit forms control specifications for a file
H	Display all FCU commands
L	Print a file's forms specifications to the current list file. Please note that if you use the L command, you must have previously set a list file or have executed FCU with the /L= switch (see FCU switches)
T	Type forms control specifications on the terminal

If you enter the C or E command, the FCU returns with an interactive question/answer dialog. Default values or current settings are enclosed in square brackets and you may select them simply by typing NEW LINE. When you change certain parameters, the system gives dependent parameters default values. If you change the line length, for example, the FCU sets default tab stops.

In all, there are 10 questions you must answer to create or edit forms control specifications. An uparrow (↑) moves you back to the previous question and a NEW LINE moves you to the next question. You may use a CTRL-C CTRL-A to interrupt the dialog and return to the COMMAND ? prompt. (The file will then be left with default form specifications if you used the C command, or with its current form specifications if you didn't use the C command.)

1. *COMMAND?*

   Type C if the file has no forms control specifications. Type E if you wish to edit a file's existing forms control specifications.

2. *PATHNAME?*

   Type the pathname or filename of the file whose forms profile you are creating or editing.

3. *CHARACTERS PER LINE (16–138) [80]?*

   Type the maximum number of characters you want on each line. If you type a NEW LINE character, you are choosing the default of 80 characters per line.

   NOTE: At print time, this number must be less than or equal to the line length of the form in the printer.

4. *TAB STOPS (2–79, OR STANDARD) [8,16,24,32,40,48,56,64,72]?*

   To set default tab stops at every eighth column (beginning at column eight), type STANDARD (or an acceptable abbreviation of STANDARD) or NEW LINE. Printing begins on the column following the tab stop. Column one and the last column are not valid tab stop positions.

5. *FORM LENGTH IN LINES PER PAGE (6–144) [66]?*

   Type the physical form's actual length. If this number is incorrect, the spooler cannot keep the paper aligned or print the file according to your specifications.

6. *TOP OF FORM (CHANNEL 1) LINE NUMBER (1–lastline) [4]?*

   Type the line number you want printing to begin on. This number also becomes the setting for Channel 1 of the VFU. The parentheses will show the number of the first line and the number of the last line you specified for printing (see question 5).

7. *BOTTOM OF FORM (CHANNEL 12) LINE NUMBER (topline–lastline) [lastline]?*

   Type the number of the last line you want to print on. This number also becomes the setting for Channel 12 of the VFU tape. The default value (shown in square brackets) is the line number you specified for the FORM LENGTH query. The top line and last line you specified are shown in parentheses.

8. *VFU TAPE (LINE NUMBERS topline–lastline, CHANNEL 2–11, OR STANDARD) [ ]?*

   To specify a null VFU tape, type STANDARD (STA) or NEW LINE. To specify a channel and line number, type the line number you want to advance to, followed by the channel number (For example, 10–2 would signify that when the line printer encounters the code for channel 2 it should advance to line 10. See Table 6-2 for a list of channel codes.) You may specify more than one line number for a channel number. In this case, when the line printer encounters the code for the channel, it advances to the next line number that you have associated with this channel.

   NOTE: Usually a NEW LINE is sufficient as a response to this question, since the default value is a null tape.

9. *OUTPUT TO PATHNAME [PATHNAME ENTERED ABOVE]?*

   Type a different pathname if you wish to copy the forms control specifications to another file. Type NEW LINE to write the current specifications to the pathname shown in square brackets.

10. *COMMAND?*

    Type any of the FCU commands.

FCU Switches

/L=list-filename Use this switch if you want to list the forms control specifications for a file (L command). If you try to list a file without appending this switch to the FCU command, or setting the CLI list file, you'll get an error message.

Argument Switches

None

Table 6-2. Vertical Forms Unit Channel Codes

CHANNEL	OCTAL CODES	ASCII CODES
1	<022> <100>	R @
2	<022> <101>	R A
3	<022> <102>	R B
4	<022> <103>	R C
5	<022> <104>	R D
6	<022> <105>	R E
7	<022> <106>	R F
8	<022> <107>	R G
9	<022> <110>	R H
10	<022> <111>	R I
11	<022> <112>	R J
12	<022> <113>	R K

In response to question 8 of the FCU dialog, you can assign values to each of the above channels (except channels 1 and 12 which are reserved for top-of-form and bottom-of-form respectively.) When the line printer encounters code for a channel, it will advance to the line you specified in question 8. Note that the parity bit is not set for the octal codes. Use the DISPLAY utility to make sure that you have the correct code in your text.

Table 6-3. Vertical Forms Unit Step Count Codes

STEP COUNT	OCTAL CODES	ASCII CODES
0	<022> <120>	R P
1	<022> <121>	R Q
2	<022> <122>	R R
3	<022> <123>	R S
4	<022> <124>	R T
5	<022> <125>	R U
6	<022> <126>	R V
7	<022> <127>	R W
8	<022> <130>	R X
9	<022> <131>	R Y
10	<022> <132>	R Z
11	<022> <133>	R [
12	<022> <134>	R \
13	<022> <135>	R]
14	<022> <136>	R ^
15	<022> <137>	R -

This table shows the effect of other ASCII codes on the VFU of the line printer. These codes cause the line printer to advance a certain number of lines relative to the current position. You cannot change these settings with the FCU.

When the line printer encounters the ASCII codes in your text it will automatically advance the number of lines specified in the step count column. Note that in the octal codes, the parity bit is not set. Use the DISPLAY utility to make sure you have the correct code in your text.

Example

```
) XEQ FCU ↵

AOS FORMS CONTROL UTILITY
REV 6.00 26-NOV-84 13:00:00

TYPE 'HELP' FOR INSTRUCTIONS

COMMAND? C ↵
PATHNAME? FILE1 ↵

CHARACTERS PER LINE (16-136)
  [80]? ↵

TAB STOPS (2-79, OR STANDARD)
  [8,16,24,32,40,48,56,64,72]
?10
?20
?30
?40
?50
? ↵

FORM LENGTH IN LINES PER PAGE (6-144)
  [66]? 60 ↵

TOP OF FORM (CHANNEL 1) LINE NUMBER (1-60)
  [4]? ↵

BOTTOM OF FORM (CHANNEL 12) LINE NUMBER
(4-60)
  [60]? ↵

VFU TAPE (LINE NUMBERS (4-60), CHANNELS
2-11, OR STANDARD)
  [ ]? ↵

OUTPUT TO PATHNAME
[:ZEB2:UDD:ZONIS:FILE2]? ↵

COMMAND? T ↵
PATHNAME? FILE1 ↵

PATHNAME
  [:UDD:ZONIS:FILE1]
CHARACTERS PER LINE
  [80]
TAB STOPS
  [10,20,30,40,50] FORM LENGTH IN LINES PER
PAGE
  [60]
TOP OF FORM (CHANNEL 1) LINE NUMBER
  [4]
BOTTOM OF FORM (CHANNEL 12) LINE NUMBER
  [60]
VFU TAPE
  [ ]

COMMAND? B ↵

FCU TERMINATING 26-NOV-84 13:05:00
)
```

In this example, user ZONIS created format specifications for FILE1. He chose the default of 80 characters per line but did not choose the default tabs. Instead, he placed tabs at 10, 20, 30, 40, and 50. He chose a form length of 60 lines per page, with printing to begin on line 4 and end on line 60 (default values). By typing a NEW LINE in response to the VFU query, he specified a standard, or null, channel. All output goes to FILE1. Finally, user ZONIS typed out the forms control specifications for FILE1 to check their accuracy before terminating the FCU.

FED *Utility*

Edit disk file locations (AOS/VS only).

Format

XEQ FED pathname

FED is the File Editor Utility, which allows you to examine and modify locations in AOS/VS disk files. (Use DEDIT for AOS disk files). For more information about FED, consult the *AOS/VS DEBUG and File Editor User's Manual* (093-000246).

FED Switches

/I=filename	Use the commands in filename for the editing session. By using this switch, you can build a file of FED commands and execute them all at once by issuing a single command. Your file must end with a BYE command.
/L=filename	Save all FED commands and responses in filename. If filename does not exist, FED creates it; if it does exist, FED appends the new information to the existing file.
/N	Do not open a symbol table (.ST) file.
/P	Treat the disk file as a program file.
/R	Open the file for read access only. Do not allow modification.
/S=filename	Use filename as the symbol table file.
/U	Treat the disk file as a user data file.
/X	Treat the disk file as an AOS/VS system file. Use this switch only to apply a patch released by Data General Corporation.

Example

```
) XEQ FED PROG1 )
FED- Disk File Editor- REV. xx
_
.
(FED commands)
.
.
_$Z
)
```

FILCOM *Utility*

Compare two files (AOS/VS only).

Format

XEQ FILCOM $pathname_1$ $pathname_2$

This utility compares $pathname_1$ to $pathname_2$. The messages that FILCOM displays when it finds a disagreement between the two files are as follows:

Number of words that disagreed = X

Number of UDA words that disagreed = X

The number of words are in decimal and the actual count replaces X.

If one file has a UDA and the other one does not the message

FILENAME does not have a UDA

appears. The actual filename replaces FILENAME. FILCOM will not display the existing UDA. You can display the UDA with the command

XEQ DISPLAY/LISTUDA

The output contains the command line echoed on the terminal screen. By default, FILCOM displays the octal value of each different word for both files unless you specify otherwise using the /RADIX switch. The output contains the address and file contents for each word that differs. FILCOM supresses any leading zeros. The display format of the last byte is <byte value>------ . The angle brackets are to remind you that the value is a byte value and not a word value.

FILCOM Switches

/ADR_RADIX=	Specifies the radix to be used when displaying the file address. Only values 2 through 16 inclusive are valid.
/HEADER	Specifies that the pathnames to the files being compared should be displayed. Header information may be extended in the future, so please do not depend on its never being modified.
/L	Write output to @LIST instead of to @OUTPUT.
/L=pathname	Write output to the file specified by pathname instead of to @OUTPUT.

FILCOM (continued)

/RADIX=	Specifies the radix to be used when displaying the file contents. Only values 2 through 16 inclusive are valid.
/UDA_ONLY	Specifies that file contents are not to be examined, only the UDA of each file is compared.

Example

The following three files are in the working directory:

FILE1	FILE2	FILE3
ABCDEFG	ABCDEFG	ABCDEFG
HIJKLMN	ABCDEFG	HIJKLM
OPQRSTU	OPQRSTU	OPQRSTU
		VWXYZ

```
) XEQ FILCOM FILE1 FILE2 ↵
FILCOM,FILE1,FILE2
    4    44111  40502
    5    45113  41504
    6    46115  42506
    7    47012  43412
Number of words (in decimal) that differed =      4
) XEQ FILCOM FILE1 FILE3 ↵
FILCOM,FILE1,FILE3
    7    47012                5117
   10    47520               50121
   11    50522               51123
   12    51524               52125
   13    52412                5126
   14    ----------          53530
   15    ----------          54532
   16    ----------  <  12>-----
Number of words (in decimal) that differed =      8
)
```

First, compares FILE1 and FILE2, then compares FILE1 and FILE3. Note that two characters are packed into each word and that the word number in the left column is in octal.

FILCOM *Utility*

Compare two files (AOS only).

Format

XEQ FILCOM $pathname_1$ $pathname_2$

This utility compares $pathname_1$ to $pathname_2$. If the files differ, FILCOM displays the word number and octal number of each different word for both files. If one file is longer than the other, the system displays all of the words in the longer file and dashes for the shorter file.

FILCOM Switches

/L	Write output to @LIST instead of to @OUTPUT.
/L=pathname	Write output to the file specified by pathname instead of to @OUTPUT.

Example

The following three files are in the working directory:

FILE1	FILE2	FILE3
ABCDEFG	ABCDEFG	ABCDEFG
HIJKLMN	ABCDEFG	HIJKLM
OPQRSTU	OPQRSTU	OPQRSTU
		VWXYZ

```
) XEQ FILCOM FILE1 FILE2 ↓
```

	FILE1	FILE2
000004	*044111*	*040502*
000005	*045113*	*041504*
000006	*046115*	*042506*
000007	*047012*	*043412*

```
) XEQ FILCOM FILE1 FILE3 ↓
```

	FILE1	FILE3
000007	*047012*	*005117*
000010	*047520*	*050121*
000011	*050522*	*051123*
000012	*051524*	*052125*
000013	*052412*	*005126*
000014	------	*053530*
000015	------	*054532*
000016	------	*0050--*

```
)
```

First, compare FILE1 and FILE2, then compare FILE1 and FILE3. Note that two characters are packed into each word and that the word number in the left column is in octal.

[!FILENAMES] *Pseudo-Macro*

Expand to a list of filenames.

Format

[!FILENAMES *[pathname]* ...]

You may use templates in the pathname argument(s). If used without arguments, this pseudo-macro expands to all filenames in the working directory (i.e., the template + is the default).

Macroname Switches

None

Argument Switches

None

Example

```
) QBATCH XEQ MASM/L/E=@LPT &↓
&) ([!FILENAMES,+.SR]) ↓
```

Create a separate batch job to assemble each file with a .SR extension in the working directory.

NOTE: If you use a template that matches a complex directory structure, this command might overflow memory. In order to avoid an overflow, you should execute !FILENAMES from LEVEL 0 with as short a command line as possible. In addition, using short directory names will permit more filenames to be handled successfully.

!FILENAMES does not resolve links. If a link file is given as the argument to the pseudo-macro, the link filename is returned, not the file specified by the link's resolution pathname. If the argument contains a link filename as a pathname element, !FILENAMES does not resolve the link, and the specified file is not found.

FILESTATUS *Command*

Display status information for one or more files.

Format

FILESTATUS *[pathname]...*

FILESTATUS lists all the specified filenames with the information requested in the switches. The default listing has a header for each directory the filenames are listed from, and lists only simple filenames.

You can use filename templates in the *pathname* argument(s). If you omit arguments, the CLI displays the names and status of all the files in the working directory.

NOTE: FILESTATUS does not resolve link files. If the command is given a link file as an argument, it returns status information for the link file itself, and not for the file specified by the link's resolution pathname. If the argument contains a link filename as a pathname element, FILESTATUS does not resolve the link, and the specified file is not found.

Use the following command switches to return information about specific files; e.g.,

) FILESTATUS/TYPE/DCR MYFILE YOURFILE)

Different forms of these switches display filenames and statistics by selected group. If a switch doesn't apply to a specific file (e.g., /ELEMENTSIZE for a directory), the system ignores it and fills its field with dashes.

Command Switches

/1= { IGNORE / WARNING / ERROR / ABORT } — Set CLASS1 to the specified severity level for this command.

/2= { IGNORE / WARNING / ERROR / ABORT } — Set CLASS2 to the specified severity level for this command.

/ASSORTMENT — Display an assortment of file information: file type, date and time of creation, and file length. If the file is a link, display file type, LNK, and link resolution name.

/DCR — Display file creation date.

/DLA — Display date file was last accessed.

/DLM — Display date file was last modified.

/ELEMENTSIZE — Display the number of disk blocks in a file element for this file. A file element is the smallest unit by which a disk file can grow (a disk block is 512 bytes).

/HASHFRAMESIZE — Display directory's hash frame size.

/INDEX — Display file's current number and maximum number of index levels.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/LENGTH — Display file's byte length.

/LINKNAME — Display link's resolution name.

/PACKET — Display the entire contents of the packet returned by the ?FSTAT system call. Refer to the AOS or AOS/VS programmer's manual for a description of this information. Most of the packet information is available through other switches.

/PERMANENCE — Display PERM if a file or directory has its permanence ON. Nothing is displayed if permanence is OFF.

/Q — Set SQUEEZE to ON for this command.

/RECORD — Display file's record format, either as an integer (fixed-length) or as one of the following mnemonics:

DYN (dynamic) — a record length is specified for each read and write.

VAR (variable) — each record starts with a header containing the size.

D-S (data sensitive) — records are terminated by specific characters embedded in the text.

/TCR — Display file creation date and time.

/TLA — Display date and time file was last accessed.

/TLM — Display date and time file was last modified.

/TYPE — Display the file's type, either as a three-character mnemonic or as a decimal number (0–255) if a mnemonic doesn't apply.

/TYPE=type — Select all files of the specified type. Types are provided in Table 2-6. type can be in the form:

m-n	decimal numbers which define a range of file types
\m-n	decimal numbers which define a range of file types to exclude
n	decimal number (0–255)
\n	decimal number to exclude
XXX	3-letter mnemonic
\XXX	3-letter mnemonic to exclude

You can use more than one /TYPE= switch in a command line.

/UDA — Display UDA if the file or directory has a User Data Area

The following switches tell the CLI to display all filenames that meet the specified conditions.

/AFTER/TLA= — List only those files that were last accessed after the specified time, date, or date:time. date:time is in the form dd-mm-yy:hh:mm:ss

/AFTER/TLM= — List only those files that were last modified after the specified time, date, or date:time. See /AFTER/TLA= for format

/BEFORE/TLA= — List only those files that were last accessed before the specified time, date, or date:time. See /AFTER/TLA= for format

/BEFORE/TLM= — List only those files that were last modified before the specified time, date, or date:time

/BEFORE/TLM=date — List only those files that were last modified before the specified date. date is in the form dd-mmm-yy

/BEFORE/TLM=date:time — List only those files that were last modified before the specified time and date. See /AFTER/TLA= for format

/BEFORE/TLM=time — List only those files that were last modified before time today. time is in the form hh:mm:ss

NOTE: You may specify a range of dates by using both the /BEFORE and /AFTER switches. However, you must use the same modifier for both: either /TLM= or /TLA=. You cannot use /TLM= and /TLA= at the same time

The following switches control formatting:

/CPL=n — Set the number of characters per line for FILESTATUS output (default is 72)

/NHEADER — Do not print directory headers. Also list files with their complete pathnames; default is directory headers

/SORT — Sort the filenames alphabetically

FILESTATUS (continued)

Argument Switches

None

Example

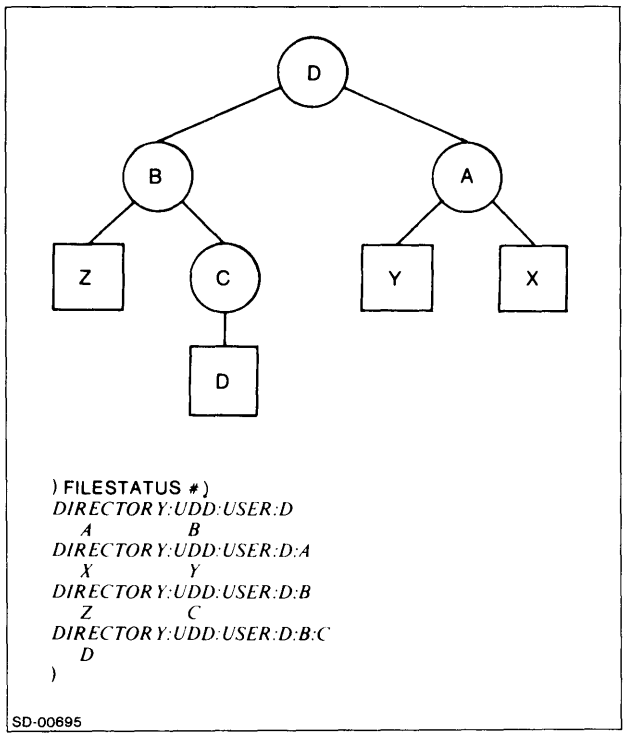
```
) FILESTATUS/ASSORTMENT/SORT ↓
DIRECTORY :UDD:USER

DDAY.DOC  TXT   6-JUN-84   6:00:00    30000
DIR1      DIR   8-DEC-83   14:39:12   123456
LINK2     LNK   :UDD:USERA
PROG.PR   PRG   24-NOV-84  20:44:26   1324

)
```

Displays a sorted list of filenames with an assortment of information including type, creation date and time, length in characters, and (for a link file) link resolution pathname.

Lists the filenames in the tree subordinate to working directory D shown in the accompanying figure.



FORT4 *Language*

Compile a FORTRAN IV source file (AOS only).

Format

```
FORT4 source-pathname
```

FORT4 is a macro that you use to compile a FORTRAN IV source file. The FORT4 command first searches for source-pathname.FR. If that is not found, it searches for source-pathname.

Output can be an assembled object file, an intermediate assembly language file, or a source listing. By default, FORT4 produces no listing, and its execution generates an intermediate source file, source-pathname.SR (compiler output), and an object file, source-pathname.OB (assembler output).

Each FORTRAN IV main program, external subroutine, and external function must be compiled separately. Once you have successfully compiled your program, use the Bind or Link utility to load it.

For a complete description of the FORTRAN IV programming language and the FORT4 command line, see the *FORTRAN IV User's Manual* (093-000053) and *FORTRAN IV Runtime Library User's Manual* (093-000142).

FORT4 Switches

/A	Abort on system error.
/B	Produce a brief listing by compiling only the source program input.
/E[=*pathname*]	Write error messages to the file specified by *pathname*. Including /E without a pathname suppresses error messages. If you omit /E, error messages are written to the current output file.
/F	Make FORTRAN variable names and statement numbers equivalent to memory locations and offsets, so that they are recognizable to the debugger.
/L	Generate a listing file and write it to the current list file.
/L=pathname	Generate a listing file and write it to the file specified by pathname.
/N	Do not produce an object file.
/NA	Compile only; do not assemble.

/O=pathname	Name the object module pathname. If you omit this switch, the object name is pathname.OB.
/P	Process only the first 72 characters of each record.
/S=pathname	Save the intermediate source file (compiler output) and name it pathname. If you use /S but omit a pathname, the source is saved and named source-pathname.SR. If you omit /S, the source file is deleted after assembly.
/U	Output user symbols in the assembly phase.
/X	Compile statements with an X in column 1.

Argument Switches

None

Example

```
) FORT4/B MYPROG ↵
```

Compiles MYPROG.FR, writing a brief listing to the current list file. Because /E is omitted, all errors are sent to @OUTPUT. The compiler produces MYPROG.OB as the object file.

```
) FORT4 MAIN ↵
) FORT4 XSUB1 ↵
) FORT4 XFUN ↵
) FORT4 XSUB2 ↵
) XEQ BIND MAIN XSUB1 XFUN XSUB2 FORT.LB ↵
```

Compiles separately the four modules that make up the complete program. Then uses the Bind utility to load the program. You must load the FORTRAN IV libraries with the program.

HELP *Command*

Explain a CLI command, pseudo-macro, or general topic.

Format

HELP *[item]...*

The CLI contains information files that you can display at your terminal by typing the HELP command. To get terse information about a command, topic or pseudo-macro name, type HELP, followed by the command, topic or pseudo-macro name. To get detailed information about a command, topic or pseudo-macro, type HELP/V, followed by the command, topic or pseudo-macro.

Some HELP file messages are longer than one screen and will scroll by quickly, unless you freeze the display. You can stop the file from scolling by too quickly by either setting the CHARACTERISTIC's /PM switch on. Or you can freeze your terminal display by using the CTRL-S control character. You would then unfreeze the display by using the CTRL-Q control character to unfreeze the display. (See the section in Chapter 1 on Control Characters.)

item can be any one of the following:

- a CLI command
- a CLI pseudo-macro
- a general CLI topic
- a utility

Command Switches

/1=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.

HELP (continued)

/Q	Set SQUEEZE to ON for this command.
/V	Display verbose description of the specified item. By default, HELP displays a terse description.

Argument Switches

None

Example

```
) HELP MOUNT
MOUNT - REQUIRES ARGUMENT(S)
   SWITCHES: /1= /2= /L(=) /Q
/DENSITY= /IBM /READONLY /VOLID=

FOR MORE HELP TYPE
HELP/V MOUNT
)
```

HELPV *Macro*

Display, in page mode, the verbose HELP command message.

Format

```
HELPV command
```

This macro displays the verbose (detailed) help message for the specified command in page mode. Page mode freezes the screen in 24-line units. To unfreeze the screen, you must use the control character CTRL-Q. You must continue to use CTRL-Q to advance the help message until it ends.

The help message appears dimmed except for command formats, command examples, and selected headings. To stop a help message display, use a CTRL-C, CTRL-A sequence. If you do this, your terminal will continue to display text dimly, and in page mode. To turn off dim mode, press the ERASE PAGE key. To turn off page mode, type CHARACTERISTICS/OFF/PM (see the CHARACTERISTICS command for more information).

HELPV Switches

None

Example

```
) HELPV FILESTATUS
   .
   .
   .
(CTRL-Q)
   .
   .
   .
(CTRL-Q)
)
```

Display the verbose (detailed) help message for the FILESTATUS command. When you have finished advancing the message with a series of CTRL-Qs, the macro turns off page mode. The last CTRL-Q returns the CLI prompt.

[!HID] *Pseudo-Macro*

Expand to a Host ID.

Format

[!HID *[hostname]*]

If no argument is given, !HID will return the ID of the local host. If you supply an argument, it must be a valid hostname. This pseudo-macro always returns a three-digit number.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE My host ID is [!HID] )
My host ID is 014
) WRITE The host ID of NET_SYS17 is & )
&) [!HID NET_SYS17] )
The host ID of NET_SYS17 is 017
)
```

HOST *Command*

Display a system's hostname.

Format

HOST [hostID]...

This command displays the hostname which corresponds with each of the *hostID* arguments. If you don't provide any arguments, HOST will display the hostname of the system on which you are running.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/S		Place the hostname in the current STRING. (See the STRING command.)

Argument Switches

None

Example

```
) HOST )
NET_SYS1
) HOST 4 )
NET_SYS4
)
```

[!HOST] *Pseudo-Macro*

Expands to a hostname.

Format

[!HOST *[hostID]*]

If you don't provide an argument, the CLI returns the local hostname. If you do supply an argument, it must be a decimal number between 1 and 127.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!HOST] )
NET_SYS1
) WRITE [!HOST 14] )
NET_SYS14
)
```

INITIALIZE *Command*

Graft a logical disk into the working directory.

Format

INITIALIZE physical-unitname *[physical-unitname...]*

This command grafts a logical disk unit (LDU), a directory made up of one or more physical disks, into the working directory.

You must name all the physical units that the volumes are mounted on. If the logical disk contains more than one physical disk, you must name each one. After executing this command, the system displays the name of the new logical disk.

Command Switches

/1=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/S		Do not display the name of the new logical disk; instead, store the name in STRING where you can use it as an argument to commands via the !STRING pseudo-macro.

Argument Switches

None

Example

```
) INITIALIZE @DPD0 ⤸
CAIN
)
```

CAIN is the highest directory on the grafted logical disk. You must have Append access to the directory onto which this LD is grafted.

```
) INITIALIZE @DPD10 @DPD14 ⤸
ADAM
)
```

The logical disk ADAM consists of two physical disk units 10 and 14.

LABEL *Utility*

Write a volume label to the beginning of a tape or diskette.

Format

```
XEQ LABEL unitname volid
```

The LABEL utility writes a label to the beginning of a magnetic tape or diskette, allowing the tape or diskette to be used for labeled access. If a tape is already labeled, relabeling effectively scratches it, preventing future access to any data already on the tape. The main benefits of labeled tape access are

- multivolume reads or writes
- later access by volume ID (volid) and filename, instead of file number

Before you can use labeled tape(s), you must use the MOUNT command to ask the system operator to mount the tapes. To request mounting, you can either use the MOUNT command or the pathname @LMT:volid:filename.

If you will be mounting your own tapes and writing your own labels and dumps, you'll have access to the system console and act as your own system operator (using CONTROL EXEC commands, for instance).

After the correct volume has been mounted, you can access the tape or diskette.

Generally, use LABEL to prepare only labeled tapes, not diskettes. For diskette labeling, the CLI's labeling feature is easier to use (as described under the OPERATOR command).

The definitions of the arguments follow:

unitname is the unit name of the tape or diskette (e.g. @MTB0, @MTC0, or @DPJ10).

volid is a volume ID, which includes from one to six filename characters. Later, for labeled access, you will use this volid.

You can check the contents of a label using the DISPLAY utility and the unit name; for example, `X DISPLAY @MTB0 ⤸`. Labels created by the LABEL utility cannot be read by the AOS MMOVE utility. (MMOVE is a utility supplied with AOS on DG DESKTOP GENERATION systems.)

For more information on labeled tape, see Chapter 7 of the *How to Generate and Run* manual for your operating system.

LABEL (continued)

LABEL Switches

/I — Create IBM labels, needed if the volume(s) will be read on a system that uses IBM-format labels. If you omit this switch, the utility creates ANSI-standard labels (suitable for DG and many other non-IBM systems). A tape cannot be relabeled without destroying all data recorded, so be sure that the destination system uses IBM format before you use this switch. Labeled tapes written in IBM format cannot be read on an ANSI-format system (and vice-versa).

/OWNER=string — Write string in the label's owner field. In an ANSI label (default), the owner field can hold 14 characters. In an IBM label, the owner field can hold 10 characters.

/S — Scratch the tape. This switch rewrites beginning and end-of-file labels for a null first file, effectively deleting all files on the tape. You must include the volume identifier (volid) with this switch to ensure that the utility is scratching the correct tape

/UVL=string — Write a user volume label of string into the label. string can be up to 76 characters. You can specify up to nine user volume labels using multiple /UVL switches.

Example

```
) XEQ LABEL @MTB0 VOL1 ↵
)
(replace tape on the unit with another)
) XEQ LABEL @MTB0 VOL2 ↵
```

These commands write the labels VOL1 and VOL2 to tapes mounted on unit MTB0. Labeling a tape effectively overwrites all files on the tapes and prepares them for labeled access.

The next command sequence asks to have the tape(s) mounted and uses them.

```
) MOUNT/VOLID=PROGS1/VOLID=PROGS2 & ↵
&) MYTAPE PLEASE ↵
```

A message from the EXEC appears on system console; the the system operator mounts tape volid PROGS1 on a tape unit, then informs EXEC where the tape is mounted.

```
) DUMP/V MYTAPE:SOURCES #:+.SR ↵
```

System dumps and verifies all files whose names end in .SR, in and below working directory, to file MYTAPE:SOURCES on the tape mounted by the operator.

When tape volume PROGS1 is used up, a message from EXEC appears on system console. The system operator mounts tape volume PROGS2 on a tape unit and informs EXEC where the tape is mounted. The dump then continues on the second volume.

```
) DISMOUNT MYTAPE THANKS ↵
)
```

A dismount message from EXEC appears on the system console; the system operator dismounts the tape and types a command that tells EXEC that the tape is dismounted.

The preceding example shows steps that occur in a multiple-volume write to labeled tape. As written, the two-volume tape file could be restored (with the working directory the same as it was originally) by the CLI commands

```
) MOUNT/VOLID=PROGS1/VOLID=PROGS2 & ↵
&) XTAPE PLEASE ↵
```

Correct tape volume is mounted and EXEC notified of unit.

```
) LOAD/V/R XTAPE:SOURCES ↵
```

(After first volume, second volume is mounted and EXEC told about unit.)

```
) DISMOUNT XTAPE ↵
```

LEVEL *Command*

Display the current CLI environment level number.

Format

LEVEL

You use this command in conjunction with the PUSH and POP commands (see Chapter 4). PUSH and POP change the current CLI environment level; LEVEL displays the current environment level number.

Command Switches

/1=	IGNORE WARNING ERROR ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE WARNING ERROR ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) LEVEL ↓
LEVEL 0
) PUSH ↓
)
) LEVEL ↓
LEVEL 1
)
```

Displays current level. Pushes a level and displays the current level again.

[!LEVEL] *Pseudo-Macro*

Expand to the current CLI environment level number.

Format

[!LEVEL]

This pseudo-macro does not accept arguments.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE THE CURRENT LEVEL IS [!LEVEL] ↓
THE CURRENT LEVEL IS 0
) PUSH ↓
) PUSH ↓
) WRITE NOW THE CURRENT LEVEL IS [!LEVEL] ↓
NOW THE CURRENT LEVEL IS 2
)
```

First, evaluates [!LEVEL] and writes the current environment level number. Then, after pushing two levels, evaluates and writes [!LEVEL] again.

LFE *Utility*

Call the Library File Editor and update library files.

Format

XEQ LFE function-letter argument *[argument]...*

This utility edits and analyzes library files, that are sets of object files with special starting and ending blocks. The extension .LB usually designates library files. See the *AOS Library File Editor User's Manual* (093-000198) or the *AOS/VS Link and Library File Editor (LFE) User's Manual* (093-000245) for further information.

LINEDIT *Utility*

Edit ASCII text (AOS only).

Format

XEQ LINEDIT *[pathname]*

This command calls the LINEDIT text editor program. If the file you specify in pathname does not exist, LINEDIT asks

DO YOU WANT THE FILE TO BE CREATED

Answer Y to create the file. LINEDIT then displays its prompt (?).

If you include *pathname* and the file exists, LINEDIT opens it for editing and displays its prompt.

If your user profile allows you to create two or more subordinate processes, you can execute CLI commands from LINEDIT. See the *AOS LINEDIT Text Editor User's Manual,* (093-000218) for more information.

LINK *Utility*

Link object modules to form an executable program file.

Format

XEQ LINK objectmodule *[argument]...*

The Link utility creates executable program files from one or more object and library files.

In the command line, the optional argument can be any of the following:

- An object file created by an assembler or compiler
- A library file created by the LFE utility
- The literals !*, !, *!, which begin, divide, and close an overlay area
- A symbol name modified by special switches

For more information about the Link utility, see either the *Advanced Operating System/Virtual Storage (AOS/VS) Link and Library File Editor (LFE) User's Manual* (093-000245) or the *Advanced Operating System (AOS) Link User's Manual* (093-000254).

LINK Switches

/ALPHABETIC	List all symbols, sorted alphabetically, and write the list to the list file. /L must accompany this switch.
/BUILDSYSTEM	Build an AOS program file.
/CHANNELS=integer	Create entry symbol ?CHAN having value integer; if /SYS=RDOS, define the maximum number of channels the program will open at one time.
/COMOVR	Place common areas within the overlays in which they were first defined.
/DEBUG	Create the undefined external symbol DEBUG, and optionally generate .DL or .DS output files.
/E=pathname	Write error messages to pathname, instead of to @OUTPUT.
/ELEMENTSIZE=n	Create program and overlay files with an element size of n.
/HEXADECIMAL	Convert octal output values to hexadecimal.
/KTOP=integer	Set the output program file's address space to integer number of 1K pages.
/L	Write a listing to the current list file.
/L=pathname	Write a listing to the file specified by pathname.
/LOCAL	Add all local symbols to symbol table.
/MAP	Write a map of all partitions, common areas, and overlays to the list file. /L must accompany this switch.
/MODMAP	Produce a listing of each module's contributions to all partitions. /L must accompany this switch.
/MODSYM	Produce a listing of each module's entry symbols. /L must accompany this switch.
/MTOP=integer	(AOS/VS only) Set the output program file's address space to integer number of megabytes.
/N	Do not create any output files, except the error and listing files.
/NBOT=integer	Set the lowest NREL address to integer.
/NCR	Remove all resource calls.
/NRP	Optimize some resource calls.
/NSLS	Suppress the scan of the system library, which by default occurs at the end of the first pass.
/NTOP=integer	Set the highest NREL address to integer.
/NUMERIC	List all symbols, sorted numerically, and write the list to the list file. /L must accompany this switch.

LINK (continued)

/O=pathname	Give all output files, except the listing and error files, the name **pathname** plus the appropriate extension.
/OBPRINT	Dump the contents of each object block. /L must accompany this switch.
/OVERWRITE	Suppress the error message when overwriting occurs.
/PADDING=n	(AOS/VS only) Pad program and overlay file lengths to the next second number boundary.
/PRSYM	Place an RDOS-style radix-50 symbol table in the program file.
/REVISION=number	Set the revision number of the program file to the specified value. Under AOS/VS, the format for **number** is **ww**[.*xx*[.*yy*[.*zz*]]]. Under AOS, the format is **ww**[.*xx*].
/RING=integer	(AOS/VS only) Set the ring of the program file to **integer**, which is in the range 0 through 7.
/SBOT=n	Set the lowest address of shared pages.
/SRES=integer	Reserve **integer** number of pages of reserved pages for ?SPAGE I/O.
/STACKSIZE=integer	Set the default stack size to **integer** number of words.
/STATISTICS	List LINK runtime statistics.
/SUPST	Do not generate an output .ST (symbol table) file.
/SYS=string	Specify the operating system for which the program file should be built. Under AOS and AOS/VS, the following are legal values for **string**: "RTOS", "RDOS", and "AOS". In addition, the following are legal only under AOS/VS: "VS16" and "VS32".
/TASKS=integer	Set the maximum number of tasks that the program file needs to execute to **integer**.
/TEMP=pathname	Place Link's temporary files in **pathname**.
/UDF	Build a user data file (UDF). UDFs are not executable, they lack system tables and system library modules.
/ULAST =partition-name	Place the partition identified by **partition-name** in the highest unshared portion of the program file, just before the default stack.
/UNUSEDSIZE=n	Set the lowest address of shared pages to NMAX + n.
/UNX	Make program file type UNX.
/V	List the input file pathnames on the listing.
/WRL	Optimize some resource calls. (See the LINK manual for more details.)
/XREF	Produce a global symbol cross-reference.
/ZBOT=integer	Set the lowest ZREL address to **integer**.
{/ZR, /UC, /UD, /SC, /SD} = {ZR, UC, UD, SC, SD}	Append the contents of the default partition on the left of the equal sign to the default partition on the right. The switches are mnemonic codes with these meanings: Zero-Relocatable, Unshared Code, Unshared Data, Shared Code, and Shared Data. You may use these in any of 16 combinations. If you equate a partition attribute with itself (e.g., /UC=UC), Link issues a warning and ignores the switch.

Argument Switches

/ALIGNMENT=integer — Align the contents of this partition on a boundary of 2 raised to the power of integer, where integer is a decimal value in the range 1 through 10. You may append this switch to a left overlay delimiter (!*), to a labeled common symbol, or to a normal partition

/CLI — Read arguments from filename.

/DEBUG — Include .DS and/or .DR data from filename.

/FORCE — Force loading of all modules in a library file.

/LOCAL — Add this module's local symbols to the symbol table. You may append this switch to an object or library filename

/MAIN — Create entry symbol .MAIN, whose value is the starting address of this module. You may append this switch to an object or library filename

/MULTIPLE — Make multiple passes over a library until no additional modules can be loaded from it.

/MULT=integer — Give the overlay area integer number of basic areas. You may append this switch to the left overlay delimiter (!*)

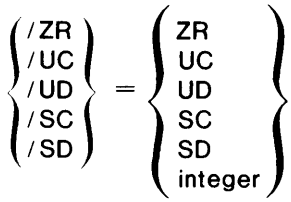
/OVERWRITE — Suppress the overwrite error message for the duration of this module. You may append this switch to a left overlay delimiter (!*), or to an object or library filename

/SHARED — Place this partition or common area in the shared area. You may append this switch to a symbol name

/START — Use this module's start address as the program's start address. You may append this switch to an object or library filename

/VAL=integer — Create this accumulating symbol and initialize it to integer. You may append this switch to a symbol name

/VIRTUAL — Create an RDOS virtual overlay. You may append this switch to a left overlay delimiter (!*)



Append the contents of the default partition on the left of the equal sign to the default partition on the right, for the duration of this module. You may use these integer values in any of 30 combinations, including one that equates a partition attribute with itself (e.g., /UC=UC). You might use such a combination to override a global switch. You may also set any partition to an integer value, which will cause that module's contribution to the given partition to be loaded absolutely at integer address. You may append this switch to a left overlay delimiter (!*), or to an object or library filename

Example

```
) XEQ LINK/L=@LPT/OVER/ULAST=UC &)
&) /NSLS/O=MY_PROG ALPHA BETA )
```

Links two modules, ALPHA and BETA, and names the output files MY_PROG.extension. /L=@LPT generates a listing and sends it to the lineprinter. /OVER suppresses overwrite messages, and /ULAST=UC places the contents of partition UC last in the unshared area of memory. /NSLS suppresses the search of the system library.

```
) XEQ LINK ALPHA BOOT/VAL=2 BETA !* &)
&) GAMMA DELTA ! RHO *! )
```

Binds five modules: ALPHA, BETA, GAMMA, DELTA, and RHO. The overlay designators specify an overlay area with two overlays, one consisting of GAMMA and DELTA, the other consisting of RHO. BOOT/VAL=2 creates the accumulating symbol BOOT and initializes its value to 2.

LISTFILE *Command*

Set or display the current LISTFILE.

Format

LISTFILE *[pathname]*

If the file specified by *pathname* doesn't exist, LISTFILE creates it. If that file does exist, LISTFILE appends the subsequent data to LISTFILE. The LISTFILE *pathname* is passed to any process created by an EXECUTE, XEQ, or DEBUG command.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/G		Set list file to the generic @LIST file (no arguments allowed). See Chapter 2 for generic filenames.
/K		Set list file to null (no arguments allowed).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set list file to the previous environment's list file (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) LISTFILE )
@LIST
) LISTFILE :UDD:PHIL:LIST )
)
```

First, displays current list file, which is the generic @LIST file. Then, sets list file to the file LIST.

[!LISTFILE] *Pseudo-Macro*

Expand to the pathname of the current LISTFILE.

Format

[!LISTFILE]

This pseudo-macro doesn't accept arguments.

Macroname Switches

/P Expand to the previous environment's list file pathname.

Argument Switches

None

Example

```
) WRITE THE CURRENT LIST FILE IS [!LISTFILE] )
THE CURRENT LIST FILE IS @LIST
) PUSH )
) LISTFILE :UDD:USER:WORK )
) WRITE NOW THE CURRENT LIST FILE IS& )
&)□ [!LISTFILE] )
NOW THE CURRENT LIST FILE IS
:UDD:USER:WORK
) WRITE [!LISTFILE/P] )
@LIST
)
```

First, evaluates [!LISTFILE] and writes the current list file, which is the generic @LIST. Then changes environment, and sets a new list file for the new environment. Evaluates and writes [!LISTFILE] for the current environment, and then, using the /P switch, for the previous environment.

LOAD *Command*

Load one or more previously dumped files into the working directory.

Format

LOAD dumpfile *[source-pathname]...*

Unless you include the /FLAT switch, LOAD will maintain dumpfile's directory tree structure in the working directory. You may use templates for the *source-pathname* argument(s). If you supply no *source-pathname* arguments, the default is the template #.

NOTE: When files are loaded from the dump file, the date and time last modified appear as the current date and time. Zero length files are, however, an exception. Because nothing is ever written to the file, the date and time last modified will remain as they were when the file was dumped.

Command Switches

/1= {IGNORE | WARNING | ERROR | ABORT}
Set CLASS1 to the specified severity level for this command.

/2= {IGNORE | WARNING | ERROR | ABORT}
Set CLASS2 to the specified severity level for this command.

/BUFFERSIZE=bytes
The maximum blocksize of the tape is bytes.

/DELETE
Delete any existing nondirectory file with the same name as a file in the dump file and load the file from the new dump file.

/DENSITY=mode
Control the magnetic density of your load tape. Use this command with variable-density units, (like MTBs). The following are your mode options:

Mode	Density
800	800 BPI
1600	1600 BPI
6250	6250 BPI (AOS/VS only)
ADM	Automatic Density Matching

LOAD (continued)

/FLAT — Do not maintain tree structure; load all files into the working directory.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=@LPT — List directory and file names of loaded files to the printer.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/N — Do not load files. Print dumped files' date and filenames only.

/NACL — Give loaded files the default ACL.

/Q — Set SQUEEZE to ON for this command.

/RECENT — Do not load if nondirectory file on disk is newer than file in dump file.

/SPECIFIC=volid — Request a specific tape volume to be loaded. EXEC will display a message on the system console that requests the tape labeled volid be mounted. After this volid is mounted, the system will execute the LOAD command on it. Normally, the tapes of a multivolume labeled tape dump must be read in the order they were dumped. This switch saves time when you want to read a tape that is in the second or subsequent volume in the dump (it eliminates the read of all previous volumes).

/TYPE=type — Select all files of the specified type. Types are provided in Table 2-6. type can be in the form:

- XXX — 3-letter mnemonic
- n — decimal number (0, 10–13, or 64–255)
- m-n — decimal numbers which define a range of file types
- \n — decimal number to exclude
- \m-n — decimal numbers which define a range of file types to exclude

You can use more than one /TYPE= switch in a command line.

/V — Verify each loaded file on @OUTPUT. When used with /DELETE or /RECENT, the /V switch also verifies each deletion.

The following switches tell the CLI to LOAD all filenames that meet the specified conditions.

/AFTER/TLA= — Load only those files that were last accessed after the specified time, date, or date:time. date:time is in the form dd-mm-yy:hh:mm:ss.

/AFTER/TLM= — Load only those files that were last modified after the specified time, date, or date:time. See /AFTER/TLA= for format.

/BEFORE/TLA= — Load only those files that were last accessed before the specified time, date, or date:time. See /AFTER/TLA= for format.

/BEFORE/TLM= — Load only those files that were last modified before the specified time, date, or date:time.

/BEFORE/TLM=date — Load only those files that were last modified before the specified date. date is in the form dd-mmm-yy.

/BEFORE/TLM=date:time — Load only those files that were last modified before the specified time and date. See /AFTER/TLA= for format.

/BEFORE/TLM=time — Load only those files that were last modified today before time. time is in the form hh:mm:ss.

NOTE: You may specify a range of dates by using both the /BEFORE and /AFTER switches. However, you must use the same modifier for both: either /TLM= or /TLA=. You cannot use /TLM= and /TLA= at the same time.

Argument Switches

None

Example

```
) LOAD/D/V @MTB0:0 ↓
DELETED MYFILE.SR
MYFILE.SR
MYFILE.PR
ZFILE.CLI
DELETED OBS.SR
OBS.SR
)
```

Loads all files that are contained in the first file of the magnetic tape mounted on unit 0 into the working directory and lists their names on the terminal. If a file in the working directory has the same name as a file in the dump file, deletes the file in the working directory and loads the file from the dump file. Also, verifies all such deletions.

```
) LOAD/RECENT @MTB0:1 +.SR ↓
)
```

Loads into the working directory each file from @MTB0:1 that ends in the extension .SR and is newer than any file with the same name.

```
)LOAD/V/SPECIFIC @LMT:VOL4:FILE0 ↓
)
```

This command loads FILE0 from volid VOL4. The /SPECIFIC switch indicates that VOL4 is not the first in the file set.

LOAD_II *Utility*

Load one or more previously dumped files into the working directory (AOS/VS only).

Format

LOAD_II dumpfile *[source-pathname]...*

The LOAD_II utility does just what the LOAD command does, but faster. All of the CLI LOAD command switches are supported. The only difference between LOAD_II and LOAD is that you type LOAD_II instead of LOAD.

The LOAD_II utility allows you to restore files dumped to tape or disk with either the DUMP_II utility or the CLI DUMP command.

When you reload a file, the pathname must match the template with which the file was dumped. Thus, if you dump A:+, to retrieve a file J in directory A, you must type LOAD_II DUMPFILE A:J. If you dumped A:+ with the flat switch, you must type LOAD_II DUMPFILE J.

Command Switches

Switches are identical to LOAD's, with the following additions:

/CONFIRM	Prompt for confirmation before attempting to delete any file. Use with the /DELETE or /RECENT switches.
/DPERMANENT	Use with /DELETE or /RECENT. Existing file of the same name will be deleted even if the permanance attribute is ON.
/ELEMENTSIZE=	Override the elementsize given for all files in the dumpfile. Note that if you use this switch to change the element size on files that will be used with shared page I/O, the new element size should be 4 or a multiple of 4. Otherwise, the ?SOPEN system call will fail.

LOAD_II (continued)

Argument Switches

None

Example

```
) LOAD_II/D/V @MTB0:0 ⤸
DELETED MYFILE.SR
MYFILE.SR
MYFILE.PR
ZFILE.CLI
DELETED OBS.SR
OBS.SR
)
```

Loads all files that are contained in the first file of the magnetic tape mounted on unit 0 into the working directory and lists their names on the terminal. If a file in the working directory has the same name as a file in the dump file, deletes the file in the working directory and loads the file from the dump file. Also, verifies all such deletions.

```
)LOAD_II/RECENT @MTB0:1 +.SR ⤸
)
```

Loads into the working directory each file from @MTB0:1 that ends in the extension .SR and is newer than any file with the same name.

LOGEVENT *Command*

Enter a message in the SYSLOG (system log) file.

Format

```
LOGEVENT message
```

This command enters the message you specify into the system log file, SYSLOG. You must be in Superuser mode (SUPERUSER on) to execute this command.

Command Switches

/1=	IGNORE, WARNING, ERROR, ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE, WARNING, ERROR, ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
)SUPERUSER ON ⤸
*)
*)LOGEVENT BOOTED NEW SYSTEM ⤸
*)
```

This sends the message *BOOTED NEW SYSTEM* to the system log event file. (We turned SUPERUSER on before issuing the command.) Note that the REPORT utility will display only the first 55 characters of the message.

LOGFILE *Command*

Set or display the current log file.

Format

LOGFILE *[pathname]*

If the file you specify in *pathname* does not exist, LOGFILE creates it. If it already exists, the CLI appends subsequent data to it.

Command Switches

/1=	IGNORE, WARNING, ERROR, ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE, WARNING, ERROR, ABORT	Set CLASS2 to the specified severity level for this command.
/K		Set the log file to null (no arguments allowed).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set current log file to the previous environment's log file.
/Q		Set SQUEEZE to ON for this command.
/V		Verify the log file to @OUTPUT.

Argument Switches

None

Example

```
) LOGFILE ↓
) LOGFILE/V FILE.LOG ↓
:UDD:JAMES:FILE.LOG
)
```

First display the current log file, which is null, and then set the log file to :UDD:JAMES:FILE.LOG.

[!LOGON] *Pseudo-Macro*

Determine if you are logged on under the EXEC and, if so, expand to CONSOLE or BATCH.

Format

[!LOGON]

This pseudo-macro does not accept arguments. If you're not sure whether you're logged on under the EXEC, enter this pseudo-macro. It will return *CONSOLE* or *BATCH* (if you're logged on in a batch stream) or nothing at all (if you aren't under the EXEC).

!LOGON is most useful in macros. See the example below.

Macroname Switches

None

Argument Switches

None

Example

Given a macro containing:

```
.
.
.
[!EQUAL,[!LOGON],BATCH]
```

Assume that you want a macro to be interactive unless it is in a batch job. You could use the example above in the macro to find out if the process is logged on in a batch stream.

MASM *Language*

Assemble one or more source files to produce an object file.

Format

XEQ MASM source-pathname *[source-pathname]...*

There are two primary macroassembler (MASM) utilities: one for AOS, and one for AOS/VS. AOS MASM assembles one or more 16-bit (AOS) assembly language source files to produce an object file compatible with the AOS system. AOS/VS MASM assembles one or more 32-bit (AOS/VS) source files to produce an object file compatible with AOS/VS. Output from both utilities can consist of object files, listing files, or both.

In order to assemble object files for your system, you must use the appropriate macroassembler. For more information about the MASM utilities and their switches, consult the *AOS Macroassembler (MASM) Reference Manual* (093-000192) or the *AOS/VS Macroassembler (MASM) Reference Manual* (093-000242).

NOTE: If you want to assemble 16-bit modules for use on AOS/VS, you must use the 16-bit MASM designed for AOS/VS. (This is known as MASM16.PR on the AOS/VS system.) Refer to the description of the MASM16 utility in this chapter.

All switches oelow apply to AOS MASM and AOS/VS MASM16 unless indicated: A few switches are for AOS/VS MASM only.

MASM Switches

/8	Use the first eight characters of symbols, not just the first five characters used for compatibility with AOS/VS MASM16 and AOS MASM, and ignored.
/16	Generate a 16-bit .OB file.
/$	Expand $ to a unique macro number. $ is normally treated as a symbol name character.
/B=pathname	Name the object file pathname.OB, instead of the name of the first source file. The default is the first argument name, with the .OB extension. This switch also overrides the .OB pseudo-op.
/CPL=integer	(AOS/VS only) Place integer characters on each line of the various output listings where integer is in the range from 80 to 136. The default is 80.
/E	Write Pass 2 error messages to @OUTPUT only if there is no listing file. Some Pass 1 error messages are automatically written to @OUTPUT.
/E=pathname	Append the error messages to pathname rather than to @OUTPUT. By default, errors are appended to @OUTPUT. If pathname is not given, and /L is given, there is no error output.
/F	Generate or suppress a form feed character as necessary to produce an even number of listing pages. By default, a form feed is generated after each page. /L must occompany this switch.
/FF	(AOS/VS only) See /F.
/K	Keep the assembler's symbol file table (MASM.ST) after the assembly is completed. By default, the symbol table is deleted.
/L	Write a listing to the current list file.
/L=pathname	Write a listing to the file specified by pathname.

/LOCAL	(AOS/VS only) Include user symbols in the object file.
/LPP=integer	(AOS/VS only) Place integer lines on each page of the various output listings, where integer is in the range from 6 to 144. /L must accompany this switch.
/M	Flag symbol redefinition statements as errors. By default, user symbols may be redefined without error.
/MAKEPS	(AOS/VS only) Skip Pass 2, do not produce an object file, and save the temporary symbol table in MASM.PS.
/MULT	(AOS/VS only) See /M.
/N	Do not generate an object file
/NOPS	(AOS/VS only) Do not use a permanent symbol table to resolve symbols in the source module.
/O	Override all listing suppression controls
/O=pathname	(AOS/VS only) Name the object file pathname.OB, rather than the name of the first source file.
/P	Add semipermanent symbols to the cross-reference
/PS=pathname	Use pathname, rather than MASM.PS, as the permanent symbol table for the current assembly.
/R	Produce an .OB file even if there are assembly errors in the source files. By default, when there are errors, MASM produces no .OB file
/S	Skip Pass 2, and save a version of the symbol table and macro definitions in MASM.PS
/STATISTICS	(AOS/VS only) Include assembly statistics in the listing. /L must accompany this switch.
/SYMBOLS=integer	(AOS/VS only) Use the first integer characters. integer can be between 5 and 32. The default is 8. /MAKEPS or /NOPS must accompany this switch.
/U	Include user symbols in the object output
/ULC	(AOS/VS only) Distinguish between upper- and lowercase letters. By default, MASM converts all symbol names to uppercase.
/XPAND	(AOS/VS only) List all source lines, regardless of listing suppression directives appearing in source. /L must accompany this switch.
/XREF=NONE /XREF=USERSYMBOLS /XREF=ALL	(AOS/VS only) Include or suppress symbols in the cross-reference listing. If you specify /L also, the default is USERSYMBOLS. If you don't specify /L, the macroassembler includes all symbols in the cross-reference.
/Z	(AOS/VS only) Begin each listing page with DATA GENERAL license notice.

Argument Switches

/PASS1	(AOS/VS only) Do not process this source file on assembly pass 2.
/S	Skip this file on Pass 2 of the assembly.

Example

```
) XEQ MASM/L=LFILE MAINPROG SUBR1& )
)& SUBR2 )
```

Assemble three modules — MAINPROG, SUBR1, and SUBR2 — and produce the object file MAINPROG.OB. Also, write a listing to the file named LFILE.

MASM16 *Language*

Assemble one or more 16-bit source files on an AOS/VS system.

Format

XEQ MASM16 pathname *[pathname]*

You use MASM16 on AOS/VS to assemble 16-bit source files. MASM16 runs under AOS/VS, but creates object files that you can use to produce program files for both AOS and AOS/VS. You can find descriptions of MASM16 switches under the MASM utility in this chapter. Note that MASM16 uses MASM16.PS, not MASM.PS. Refer to the *AOS Macroassembler (MASM) Reference Manual* (093-000192) for a complete description of 16-bit MASM and its switches.

MESSAGE *Command*

Display text message corresponding to error code arguments.

Format

MESSAGE errorcode *[errorcode]...*

This command looks up the text for error codes in the error message file and displays it on @OUTPUT (if you do not use the /L switch) or on @LIST (if you use the /L switch).

Command Switches

/1=	IGNORE WARNING ERROR ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE WARNING ERROR ABORT	Set CLASS2 to the specified severity level for this command.
/D		Accept arguments as decimal integers; default is octal integers.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) MESSAGE 25 )
25 FILE DOES NOT EXIST
)
```

Display the error message for error code 25.

MKABS *Utility*

Convert an RDOS save file to an absolute binary file (AOS only).

Format

XEQ MKABS $filename_1$ $filename_2$

This utility runs on both AOS and AOS/VS systems and produces an absolute binary file which will run on a stand-alone system. $filename_1$ is a LINK save file that you want to input to MKABS. $filename_2$ is a file you designate to receive output. MKABS outputs $filename_1$ to $filename_2$ in absolute binary form. You can then use the binary loader to load the absolute binary file onto a stand-alone system, and execute the program.

MKABS Switches

/FROM=N — Set the first save file address that MKABS will include in absolute file output equal to N. MKABS evaluates N as an octal number, relative to location zero. If you omit this switch, MKABS uses the first save file address (0 for RTOS files, 168 for RDOS files) as the from address.

/LAST=N — Set the last save file address that MKABS will include in absolute file output equal to N. MKABS evaluates N as an octal number, relative to location 0. If you omit this switch, MKABS uses the final save file address as the last address of that file.

/START=N — Set the second word in the absolute binary file's start block to starting address N, where N is an octal number in the range $-1 < N < 77777$. After the file is loaded, the program begins execution at the starting address. If N is negative, the loader will halt after loading. If N is omitted, the system uses the address specified in the RDOS user table (USTSA) as the start address. If LINK detected no starting address when the file was created, USTSA will contain a -1, and the loader will halt after loading.

/ZERO — Assume that the save file begins at core image location zero and that it was developed for RTOS. If you omit this switch, MKABS assumes that the save file begins at location 168 and that it was developed for RDOS.

Argument Switches

None

Example

```
) XEQ MKABS/START=300 RDOS.SV ABIN ↵
```

Copy the program contained in RDOS save file RDOS.SV into file ABIN; copy it in absolute binary form. After you use the basic binary loader to load ABIN onto a stand-alone system, it will begin executing at starting address 300_8.

MMOVE *Utility*

Transfers files to and from multiple diskettes (AOS only).

Format

MMOVE diskette_name *[filename...]*

diskette_name specifies the name of a diskette drive, such as @DPM0.

filename specifies one or more files subordinate to the working directory. You may specify any number of filenames and/or templates. If you do not specify any filenames, the utility assumes the default template of #; i.e., all files in the working directory and any of its subdirectories.

If you are dumping files to the diskette, MMOVE transfers files until the specified diskette is full and then asks for another diskette. As each diskette is filled, MMOVE displays which number in the dump sequence has been assigned to the diskette. If you are loading files from the diskette, MMOVE transfers files until the diskette is exhausted and then prompts you to replace the diskette with the next one in the sequence. During a load, if you insert a diskette out of sequence or a diskette from another dump, MMOVE continues to prompt you for the proper diskette until you mount that diskette.

MMOVE supports all templates except the = for the current directory. This restricts the use of MMOVE with an argument list such as [!F, +.CLI] which would expand to =+.CLI.

NOTE: Because of dump format differences, AOS files dumped with AOS MMOVE cannot be loaded with MP/AOS MOVE or MP/OS MOVE.

MMOVE Switches

/DELETE	Delete any file in the destination directory with the same name as the file being transferred.
/DUMP	Transfer specified files to diskette.
/DUMP/FROM	Transfer specified files from diskette. /FROM can only be used in conjunction with /DUMP.
/L	Send to the line printer a list of all the files transferred.
/L=filename	Send to the specified file a list of all the files transferred.
/RECENT	If there is a file in the destination directory with the same name as a filename, transfer that filename only if it is more recent than the existing file.
/V	Displays the name of each file transferred.

The following switches tell the CLI to display all filenames that meet the specified conditions.

/AFTER/TLM=	Transfer only those files that were last modified before the specified time, date, or date:time date is in the form dd-mmm-yy, time is in the form hh:mm:ss, and date:time is in the form dd-mmm-yy:hh:mm:ss.
/BEFORE/TLM=	Transfer only those files that were last modified before the specified time, date, or date:time date is in the form dd-mmm-yy, time is in the form hh:mm:ss, and date:time is in the form dd-mmm-yy:hh:mm:ss.

Argument Switches

None

Example

) X MMOVE/DUMP/L @DPM0 ⤸

Transfers all files in your working directory (and subordinate subdirectories) to the diskette in drive @DPM0. /L produces a line printer listing of all the files transferred. When the diskette is full, MMOVE requests you to supply another diskette by displaying the following message:

Diskette 1 is full. Mount next diskette and type NEWLINE to continue.

The process continues until all files in your directory and subdirectories have been transferred to diskette.

) X MMOVE/DUMP/FROM/V @DPM0 ⤸

Transfers all files from the diskette in drive @DPM0 and subsequent diskettes (if any) into your working directory. The /V switch causes each filename to be displayed as it is transferred. When all the files on the diskette have been transferred, you are asked to place the next diskette in the drive. The following message is displayed:

This diskette is exhausted. Mount diskette 2 and type NEWLINE to continue.

MMOVE verifies that subsequent diskettes are from the same dump and also verifies their order. If you supply the wrong diskette, the following message is displayed:

Wrong diskette. Please mount diskette 2 and type NEWLINE to continue.

This is repeated until all the files from all of your diskettes have been transferred.

MOUNT *Command*

Via EXEC ask the system operator to physically mount one or more tapes.

Format

MOUNT linkname message

The MOUNT command posts a mount request on the system console. It's most often used to request the mounting of a multiple volume labeled tape fileset. You can use MOUNT for a single labeled tape or unlabeled tape volume also. MOUNT is useful with tape only; for labeled diskettes, see the OPERATOR command.

linkname is a filename that you want to use for tape access. It can be any valid filename and need have no relation to any file on the tape(s). This linkname is created in your initial user directory (form :UDD:username:linkname). It cannot already exist in your initial working diriectory. You must use the linkname — not the tape device name — for access. The link will be deleted when you use the DISMOUNT command to ask that the tape be dismounted.

message is the text you want displayed on the operator's terminal. It can be a single character or many characters up to 80. Use as many characters as needed to enable the system operator to find and mount the tape.

If you will be mounting your own tapes and performing your own loads and dumps, you'll have access to the system console and act as your own system operator (using CONTROL EXEC commands, for instance).

The MOUNT command won't work unless a system operator is on duty (someone has typed CONTROL @EXEC OPERATOR ON ⤸ on the system console). You can find out if the operator is on duty by typing

WRITE [!OPERATOR] ⤸

After you type a valid MOUNT command, the CLI prompt doesn't return until the operator responds. You can abort the mount request, if desired, with CTRL-C CTRL-A.

You can ask for a tape mount without using the MOUNT command. If you type a command that includes the form @LMT:username:filename, for example

) LOAD/V @LMT:PROGS:MYPROG+ ⤸

Then the system operator will be prompted to mount the tape that has the volume ID (volid) you specify (in the example, the volid is PROGS).

You can discover the contents of a tape label, including the volume ID and filename, with the DISPLAY utility; for example, X DISPLAY @MTB0 ⤸.

MOUNT (continued)

Command Switches

/1= { IGNORE / WARNING / ERROR / ABORT }
Set CLASS1 to the specified severity level for this command.

/2= { IGNORE / WARNING / ERROR / ABORT }
Set CLASS2 to the specified severity level for this command.

/DENSITY=mode
Select the density to be used for the tape to be mounted. The mode must be possible for the type of tape unit, and it must have been enabled at system generation. All the modes are

Mode	Density
800	800 BPI (bits per inch)
1600	1600 BPI
6250	6250 BPI (AOS/VS only)
ADM	Automatic Density Matching (on reads)

/EXTEND
Extend the list of tape volids if too few tape volids were specified (/VOLID switch). The /EXTEND switch is most useful when you are writing material to tape and you may not be able to tell how many tape volumes you will need.

/L
Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname
Write CLI output to the file specified by pathname instead of to @OUTPUT.

/Q
Set SQUEEZE to ON for this command.

/READONLY
Ask the system operator to remove the write enable ring, and set the ACL of any tape mounted from this command to username, RE.

/VOLID=volid
Restrict the mount request to labeled tape, to the tape volume with the volid of volid (written by the LABEL program). You can specify multiple volumes with multiple /VOLID switches. EXEC will tell the operator to change volumes as needed, and EXEC will check that each tape volid is mounted in the order you specify here. For multiple volume requests, you can include all volid names, or you can use the /EXTEND switch, which allows additional volumes to be mounted when the ones you specify with the /VOLID switch have all been written to (or read from).

After the system operator types CONTROL EXEC MOUNTED, the system creates a link file in your initial user directory (:UDD:username:linkname). The link file for labeled tape resolves to @LMT:first-volid.

Argument Switches

None

Example

Here is an unlabeled tape example.

```
) MOUNT TAPE1 PLEASE MOUNT TAPE Z280 ↓
```
(Operator gets and mounts tape, then types a command that tells EXEC what unit the tape is on. Since no label is was specified (no /VOLID switch) the system cannot check that the tape is the one reqested.)

```
) DUMP/V TAPE1:3 +.SR ↓
) DISMOUNT TAPE1 ↓
```

The MOUNT command tells the operator to mount tape numbered Z280. When the operator has notified EXEC that a tape is mounted, EXEC creates the link :UDD:username:TAPE1 to the tape unit the operator specified. The DUMP command then dumps all files ending in .SR to tape file 3 on this unit. When the dump is done, the DISMOUNT command asks the operator to dismount the tape.

The next command example shows a multivolume labeled tape mount request.

```
) MOUNT/VOLID=VOL1/VOLID=VOL2& ↓
&) /VOLID=VOL3/VOLID=/VOL4/EXTEND & ↓
&) MYTAPE PLEASE ↓
```

This MOUNT command asks the system operator (by using EXEC) to mount tape volumes VOL1, VOL2, VOL3, and VOL4, in that order, on an available tape unit. (EXEC must be running when the operator mounts tapes.) The /EXTEND switch allows the volid list to be extended if all material to be written won't fit on the four tapes. When the first volume has been mounted and the unit identified to EXEC, the CLI prompt returns to the user console. The user can then use the labeled tape set as desired — probably to dump or copy a very large file.

For other examples, see the DISMOUNT command or the *How to Generate and Run* manual for your operating system.

MOVE *Command*

Move copies of one or more files.

Format

MOVE dest-dir *[sourcefile]...*

This command moves copies of one or more files to the destination-directory, **dest-dir**. Normally, all *sourcefiles* must be inferior to the working directory. You may use templates for the *sourcefile* argument(s). If you supply no *sourcefile,* the template # is assumed; this means that all subordinate directories and files will also be moved, retaining the current tree structure. To move directories and files beneath the working directory without retaining the current tree structure, use the /FLAT switch (see example).

You may use filename templates for *sourcefiles* but not for the **dest-dir**.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/BACKUP		Use checkpointing when moving files with FTA. If the file transfer is aborted, a UID will be returned for use with /BACKUP=UID.
/BACKUP=UID		Use a UID returned by FTA for recovering from an aborted file transfer for which the /BACKUP switch was specified.
/BUFFERSIZE=n		Read the sourcefile(s) into a buffer of size n bytes.
/COMPRESS		Use compression when moving files with FTA.
/DELETE		Delete the nondirectory file in the dest-dir if it has the same filename as a source file.

MOVE (continued)

/FLAT — Do not maintain tree structure, move all source files into **dest-dir** (see example).

/FTA — Copy the files via FTA, XODIAC's File Transfer Agent. MOVE/FTA unlike MOVE without /FTA, resolves links; this means that it moves the link's resolution file, not the link file itself. FTA ignores the /BUFFERSIZE and /NACL switches, and does not acknowledge files deleted as a result of the /DELETE or /RECENT switches.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by **pathname** instead of to @OUTPUT.

/NACL — Give the new files the default ACL. See the DEFACL command for a discussion of default ACLs. Note that RMA ignores /NACL when moving files to a remote directory.

/Q — Set SQUEEZE to ON for this command.

/RECENT — If there is a nondirectory file in dest-dir with the same filename as a source file, move that source file only if it is more recent than the existing file.

/TYPE=type — Select all files of the specified **type**. Types are provided in Table 2-6. **type** can be in the form:

- **XXX** — 3-letter mnemonic
- **n** — decimal number (0, 10–13, or 64–255)
- **m-n** — decimal numbers which define a range of file types
- **\n** — decimal number to exclude
- **\m-n** — decimal numbers which define a range of files types to exclude

You can use more than one /TYPE= switch in a command line.

/V — List the name of each file the system moves on @OUTPUT. When used with /DELETE or /RECENT, the /V switch also verifies each deletion.

The following switches tell the CLI to MOVE all file that meet the specified conditions.

/AFTER/TLA= — Move only those files that were last accessed after the specified time, date, or date:time. date:time is in the form **dd-mmy:hh:mm:ss**.

/AFTER/TLM= — Move only those files that were last modified after the specified time, date, or date:time. See /AFTER/TLA for format.

/BEFORE/TLA= — Move only those files that were last accessed before the specified time, date, or date:time. See /AFTER/TLA for format.

/BEFORE/TLM= — Move only those files that were last modified before the specified time, date, or date:time See /AFTER/TLA= for format.

/BEFORE/TLM=date — Move only those files that were last modified before the specified date. **date** is in the form **dd:mmm:yy**.

/BEFORE/TLM =date:time — Move only those files that were last modified before the specified time and date. **date:time** is in the following form: **dd-mmm-yy:hh:mm:ss**.

/BEFORE/TLM=time — Move only those files that were last modified today before time. time is in the form hh:mm:ss.

NOTE: You may specify a range of dates by using both the /BEFORE and /AFTER switches. However, you must use the same modifier for both: either /TLM= or /TLA=. You cannot use /TLM= and /TLA= at the same time.

Argument Switches

None

Example

```
) MOVE ↑DIR1 ↓
)
```

Moves the entire subtree inferior to the working directory to DIR1, which is in the immediately superior directory. Note that since you specified no source files, the CLI assumes #.

```
) MOVE/FLAT DIR2 ↓
)
```

Moves each file that is inferior to the working directory into DIR2. Note that the system doesn't maintain the tree structure; instead, it lists each file directly in DIR2.

```
) MOVE/TYPE=64-68/TYPE=\66 &↓
&) UDD:BOB:DIR3 +↓
)
```

Moves all files of types UDF (64), PRG (65), STF (67), and TXT (68) that are contained in the working directory to DIR3. Note that files of type UPF (66) are not moved and that the sourcefile template directs the CLI to move only files from the working directory.

```
) MOVE/AFTER/TLM=1-APR-84:12:30:00 &↓
&) DIR4 +.SR ↓
)
```

Moves each file ending with the extension .SR that was last modified on or after April Fool's Day of 1984 at 12:30 p.m. and that is contained in the working directory to DIR4 (which is inferior to the current working directory).

```
) MOVE/V/DELETE DIR1 MYFILE+ ↓
DELETED MYFILE.SR
MYFILE.SR
MYFILE.CLI
)
```

Moves all files matching the MYFILE+ template to directory DIR1. If a file in DIR1 has the same filename as one of the files to be moved, deletes the file in the destination directory. Also, verifies the operation by listing the name of each file deleted or moved.

```
) MOVE/V/FTA :NET:M600_6:UDD& ↓
&) :MICHAEL:MACROS +.CLI ↓
SPEED.CLI
LINK.CLI
SETSEA.CLI
FAS.CLI
)
```

Moves all files using the File Transfer Agent ending with .CLI to the directory :UDD:MICHAEL:MACROS on a remote system. Also verifies the move by listing the name of each file moved.

MPL *Utility*

Invoke the macro processor for procedural languages (MPL).

Format

XEQ MPL inputfile-pathname

The macro processor for procedural languages (MPL) lets you define and refer to macros for use in programs written in high-level languages. MPL processes macros in a file that contains source code for the target language. Target languages are Data General's PL/I, FORTRAN 5, and DG/L languages. In addition, MPL can be used to process program text in almost any high-level procedural language.

You execute MPL before compiling a program, to translate MPL code (source code including macro definitions and references) into source code acceptable to the compiler.

MPL first looks for the input file named inputfile-pathname.MPL. If the search fails, it looks for inputfile-pathname. An output file has the appropriate language extension: .DG (for DG/L), .PL1 (for PL/I), and .FR (for FORTRAN 5). You can specify an output file with the /O switch (MPL adds the language extension if you omit it). Otherwise, MPL takes the name of the input file and adds the proper extension for the output file.

When MPL executes, it reads a file to determine which characters need special interpretation. You specify the proper file with the /LANGUAGE=language-file switch. If you omit the switch, MPL uses the default file, which is MPL.LANGUAGE. Some language files are provided with the utility. You select these by specifying the proper filename: DGL.LANGUAGE, PL1.LANGUAGE, or F5.LANGUAGE. If you have created your own language file, you use that pathname as the value for language-file. In each case, MPL must be able to access the appropriate language file or files.

You may abbreviate the switches in the command line, as long as the abbreviation remains unique.

For a complete discussion of this utility, see the *Macro Processor for Procedural Languages (MPL) User's Manual* (093-000253).

MPL Switches

/E=pathname	Write error messages to the file specified by pathname, instead of to @OUTPUT.
/ERRORLIMIT =integer	Terminate processing if the number of errors exceeds integer, where integer is in the range 0 to 32767.
/LANGUAGE =language-file	Use the special characters defined in language-file. For possible values, see the discussion above.
/N	Produce no output. /N overrides /O.
/O=pathname	Write output to the file specified by pathname.

Example

```
) XEQ MPL/LANGUAGE=PL1.LANGUAGE IN_FILE )
```

Invokes MPL to process the file named IN_FILE.MPL. MPL will use the PL/I language file, named PL1.LANGUAGE.

[!NEQUAL] *Pseudo-Macro*

Include input conditionally.

Format

[!NEQUAL, argument$_1$, argument$_2$]

This pseudo-macro begins a sequence of CLI input that the CLI is to conditionally execute. You must end the sequence with the !END pseudo-macro, and the sequence may optionally include the !ELSE pseudo-macro.

The !NEQUAL pseudo-macro must always have two arguments, which it compares character by character. If they don't match, !NEQUAL executes the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, !NEQUAL does not execute the input following it up to the !END.

If the arguments match, !NEQUAL does not execute the commands up to the !ELSE or !END. If there is an !ELSE, it executes the input following the !ELSE up to the !END.

See Chapter 5 for a discussion of conditional pseudo-macros.

Macroname Switches

None

Argument Switches

None

Example

Given a macro including:

```
[!NEQUAL,%1%,*]
WRITE NOT AN ASTERISK
[!ELSE]
WRITE AN ASTERISK
[!END]
```

This macro will write *NOT AN ASTERISK* if you call it with any argument except *; otherwise, it writes *AN ASTERISK*.

Note that you can also code the macro as follows:

```
WRITE [!NEQUAL,%1%,*] NOT AN ASTERISK &
[!ELSE] AN ASTERISK [!END]
```

Notice that we used commas to separate the arguments in the !NEQUAL pseudo-macro. If we used spaces, and argumentsub1 was null (or not present), the spaces on either side of the %1% would have become a single delimiter, giving [!NEQUAL,*]. This format is invalid, since the !NEQUAL pseudo-macro takes exactly two arguments. Notice that there are no spaces between the bracketed !NEQUAL statement and other commands and arguments.

[!OCTAL] *Pseudo-Macro*

Convert a decimal number to octal.

Format

[!OCTAL decimal-number]

The number must be a positive decimal integer in the range 0 to 4,294,967,295. The result will be in the range 0 to 37,777,777,777.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!OCTAL 1000] ↓
1750
)
```

OPERATOR *Command*

Turn on or off the CLI's ability to load from, label, and dump to labeled diskettes.

Format

OPERATOR [*ON* / *OFF*]

The OPERATOR command turns CLI operator mode on or off, or checks operator mode status. If you omit arguments, the CLI displays the current status. If you include ON, the CLI turns on operator mode (if not already on). If you include OFF, the CLI turns off operator mode (if on).

When operator mode is on, the CLI can read, write, and label diskettes. Operator mode can be turned on for any CLI process. It stays on until turned off or until the process terminates (as when you log off).

When operator mode is on, the CLI can load from, dump to, and label diskettes. Operator mode can be turned on for any CLI process. It stays on until turned off or until the process terminates (as when you log off).

When operator mode is off, the CLI cannot access a diskette by label. An attempt to do so provokes a *NO OPERATOR AVAILABLE* error message. The CLI can as usual, access a physical diskette by unit name; for example, using the DUMP or LOAD command, as in

```
) DUMP/V @DPJ10 MYFILE ↓
```

NOTE: The CLI OPERATOR command is not related to EXEC's OPERATOR command or the !OPERATOR pseudo-macro. EXEC's OPERATOR command and the !OPERATOR pseudo-macro tell whether or not anyone is on duty at the system console. The CLI OPERATOR command controls operator mode; it's available to any user. It works in any CLI process, running on any console.

Multivolume labeled diskettes are needed for backup in systems that don't have a tape unit. They are useful in any situation where someone wants to dump to diskette and the material involved exceeds the capacity of one diskette.

Whenever you do a labeled diskette dump (like a backup), be sure to write the date, volume ID (called volid), filename, and template used (if any) on the paper label of the first diskette used. Write the volid and filename on the paper label of each subsequent diskette. Doing this records needed information for access later on.

The default ACL for any diskette unit does not allow user access. If, without Superuser on, you get an *ACCESS DENIED* message, you can either turn Superuser on (if you have the privilege), or arrange to have the diskette unit ACL changed to +,WARE to allow access. Changing the unit ACL is the better course — and usually this is done in the system UP macro, with a command like

```
ACL @DPJ10 +,WARE
```

While you are loading from (or dumping to) diskette, the system does not automatically protect your diskette from access by other users. For example, user JACK can write to a diskette that user OP is trying to read. With Superuser on, or if you have owner (O) access to the diskette unit, you can change the unit ACL to prevent other users from accessing it. For example, for diskette DPJ10, you might type

```
*) ACL/V @DPJ10 ↓
@DPJ10 +,WARE
```
(It displays the ACL.)
```
*) ACL @DPJ10 JACK,OWARE ↓
```
(Set ACL for your use.)

Then use the unit. When you're finished with the unit, restore its original ACL so others can use it.

Command Switches

/1= {IGNORE | WARNING | ERROR | ABORT}
Set CLASS1 to the specified severity level for this command.

/2= {IGNORE | WARNING | ERROR | ABORT}
Set CLASS2 to the specified severity level for this command.

/L
Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname
Write CLI output to the file specified by pathname instead of to @OUTPUT.

/LABEL
On dumps only: Tells the CLI to to label diskettes without warning you before doing so. On any dump to a labeled diskette, the CLI will check the diskette before starting to write. If the diskette has the volid expected, the CLI wil start writing to it.

If the diskette does not have the volid expected, the CLI's action will depend on whether you included this /LABEL switch. If you omitted /LABEL, the CLI will display an error message and ask if you want it to relabel the diskette. If you included /LABEL (for example, OPERATOR/LABEL ON ↓), the CLI will relabel the diskette without asking for your okay.

The /LABEL switch is useful for dumps when the diskettes you want to use are not labeled, or when you don't care about their volids. It eliminates the volid specification step on a volid conflict. /LABEL is essential when you want to dump to a labeled diskette set whose retention period (default 90 days from dump) hasn't expired.

Whether or not you include /LABEL, the CLI will create volids for you as described in the next section, under "volid."

/Q
Set SQUEEZE to ON for this command.

Argument Switches

None

To access a labeled diskette, you must use the form

command @LFD:volid:filename

Where **Means**

command is the DUMP command, to write to, or the LOAD command, to read from, one or more labeled diskettes.

On a dump to a labeled diskette set, if you omitted /LABEL from the operator command, the retention period specified in the previous dump must have elapsed. If not, the CLI will report a *CANNOT DELETE UNEXPIRED FILE ON LMT* error. The default retention period is 90 days. With the DUMP /RETAIN= switch, you can specify a different period, including 0 days (which allows the diskette set to be reused immediately).

@LFD is the filename that indicates a labeled diskette (labeled floppy disk). You must include @LFD for labeled diskette access.

volid is the volid that's on the first diskette (on a load) or the volid you want written on the first diskette (on a dump). A volid can be written to a diskette either by the CLI or by the LABEL utility. The CLI's labeling mechanism is more convenient. Volids are limited to six legal filename characters.

On a load, the volid you specify must match the volid on the first diskette. If not, the CLI will signal an error when it checks the diskette label.

On a dump, the CLI checks the diskette label. If the label is correct, the dump proceeds. If the label is wrong, the CLI's message depends on whether you included the /LABEL switch when you turned operator mode on. If you omitted /LABEL, the CLI warns you that the label doesn't match the one expected. Then it gives you the choice of relabeling the diskette or inserting another diskette. If you included /LABEL, the CLI relabels the diskette with a new volid and tells you it has done so.

For access to the second and subsequent diskettes, the CLI creates default volids, based on the original volid you specified. It creates a numeric sequence by adding 1 to each volid. If the volid you originally specified doesn't end with a number, the CLI will add a number, space permitting. If there isn't room for the CLI to add a number, it will drop as many characters as needed to create the next volid. For example:

Volid you specify	Default volids created by CLI
VOL1	VOL1 (for first diskette) VOL2 (for second diskette, if needed)
	VOL10 (for tenth diskette, if needed)
BACKUP	BACKUP (for first diskette) BACKU1 (for second diskette, if needed)
	BACK10 (for tenth diskette, if needed)

filename is the filename of the diskette fileset (on a LOAD command); or it is the filename you want to create for the diskette fileset (on a DUMP command).

This filename applies to the entire fileset of one or more diskettes. The same filename used to dump to a labeled diskette set must be used to load the diskette set; if not, the CLI will report a *FILE DOES NOT EXIST* message when it checks the label of the first diskette. The filename is limited to 17 filename characters

With operator mode on, after you type a command of form @LFD:xx the CLI will prompt you to insert a diskette in a specific unit, as follows

PLEASE INSERT A DISKETTE
UNIT [@DPxn] VOLUME ID [volid] ? [Y]

The default diskette unit is @DPJ10. The displayed volid is the one you specified (first diskette) or the next sequential number after the preceding volid. If you want to override the default unit, type N ⟩. The CLI will give you a chance to specify a different default unit.

The volid and filename you specify cannot be changed during a dump or load command. To use a different volid or filename, you must abort the command and restart it. For example, assume you type

) DUMP/V @LFD:XVOL1:MYFILE ⟩

The CLI displays the *PLEASE INSERT* message shown above. But then you decide that you prefer VOL1 to XVOL1 as a first volid. To change the volid, abort the dump by typing CTRL-C CTRL-A, then retype the command with the new volid. In this case, you'd type DUMP/V @LFD:VOL1:MYFILE ⤸.

For any dump, each diskette must be hardware formatted, but need not be software formatted with the Disk Formatter. During any load operation, the first diskette must have the volid you specify, and each of the following diskettes must have the volid that the CLI expects. If you make a mistake, like typing the wrong volid or filename, abort the command via CTRL-C CTRL-A and restart it, as shown above.

For specifics on using OPERATOR and labeled diskettes for backup, see Chapter Seven or the *How to Generate and Run* manual for your operating system.

Examples

Here is an example of a multiple-diskette write, with diskette labeling. It is a dump of Allan's user directory.

) DIR :UDD:ALLAN ⤸
In initial user directory.

) OPERATOR/LABEL ON ⤸
Turn operator mode on, with automatic labeling.

) DUMP/V/RETAIN=0 @lfd:al1:allfiles ⤸
This dumps all files. Templates — as usual — could be used. The /RETAIN=0 switch allows the diskette set to be reused immediately.

PLEASE INSERT A DISKETTE
UNIT [@DPJ10] VOLUME ID [AL1] ? [Y] ⤸
CLI prompts for diskette. Insert one and press ⤸. If diskette is unlabeled or the wrong diskette, the CLI labels the diskette with volid AL1 and then displays the message

*** (RE)LABELED DISKETTE ** NEW VOLUME ID [AL1]*

PLEASE INSERT NEXT DISKETTE
UNIT [@DPJ10] VOLUME ID [AL2] ? [Y]
CLI prompts for next diskette, incrementing the volid.

⤸

Remove the diskette, insert a new one; press ⤸.

CLI labels the diskette if needed, dumps files and displays filenames.

) OPERATOR OFF ⤸
Dump completes; prompt returns; Allan turns operator mode off.

Allan then writes the volid, filename, and date on the first diskette, and volid and filename on subsequent diskettes.

After this sequence, Allan could restore the files from this diskette fileset using the commands

) DIR :UDD:ALLAN ⤸
Get to the directory from which the dump command was issued.

) OPERATOR ON ⤸
Turn operator mode on.

) LOAD/V/R @lfd:AL1:ALLFILES ⤸
Start the load...

A second example, in which the system operator (user OP) backs up the entire system, follows. A second example, in which the person acting as system operator (user OP) backs up the entire system, follows. In this example, the diskettes have already been used for a system backup, with a retention period of 0 days (DUMP/RETAIN=0...). Thus, the person doing the backup omits the /LABEL switch, to ensure that a warning will occur if he/she accidentally inserts a diskette from that's not from this diskette set.

First, the system operator must ensure that all users have logged off and that all server processes — like CEO and INFOS II — are shut down.

) DIR : ⤸
Dir to the root directory.

) OPERATOR ON ⤸
Turn operator mode on.

) SUPERUSER ON ⤸
Superuser on to allow file access.

*) ACL/V @DPJ10 OP,OWARE ⤸
Set sole access to the diskette unit.

*) DUMP/RETAIN=0/V/L=DFILE & ⤸
&) @LFD:FULL1:BACKUP ⤸
Start the dump. The RETAIN=0 switch allows the diskette set to be reused immediately. (The default retention period is 90 days from the dump date.)

PLEASE INSERT A DISKETTE
UNIT [@DPJ10] VOLUME ID [FULL1] ? [Y] ⤸
CLI prompts for diskette. Insert one and press ⤸.

THE LABEL ON THE DISKETTE IS NOT THE LABEL REQUESTED
INSERTED: PIR23 REQUESTED: FULL1
CLI checks the label. Since auto labeling was not specified, the CLI displays an error message and the next question if a conflict occurs.

OPERATOR (continued)

DO YOU WANT TO RELABEL THIS DISKETTE? [N] ⤸
Don't want to relabel.

PLEASE INSERT A DISKETTE
UNIT [@DPJ10] VOLUME ID [FULL1] ? [Y]
Again, CLI prompts for a diskette. Remove the wrong diskette, finds the correct diskette, insert it, and press ⤸.

This time, the volid is correct. The system dumps files and writes their names to listing file DFILE.

PLEASE INSERT NEXT DISKETTE
UNIT [@DPJ10] VOLUME ID [FULL2] ? [Y]
CLI prompts for next diskette, incrementing the volid.

⤸

Remove the diskette, insert the next one; press ⤸.

Again, the CLI checks the volid. The volid is correct, so the system continues dumping files and writing their names to DFILE. The dump proceeds through other diskettes.

PLEASE INSERT NEXT DISKETTE
UNIT [@DPJ10] VOLUME ID [FULL33] ? [Y]
Once more, the CLI prompts for next diskette.

⤸

Remove the diskette, insert the next one; press ⤸.

A few files are dumped...

```
*) QPRINT DFILE ⤸
```
The full backup is done. Print listing file...

```
*) DELETE/V DFILE ⤸
```
DELETED DFILE
Delete listing file.

```
*) ACL @DPJ10 +,WARE ⤸
*)
```
Restores old ACL.

The system operator who did the dump now removes the diskette from unit DPJ10 and stores all diskettes safely. Later on, if needed, all files could be restored to the disk with the commands:

```
) DIR : ⤸
) OPERATOR ON ⤸
) SUPERUSER ON ⤸
) LOAD/V/R @LFD:FULL1:BACKUP ⤸
```

[!OPERATOR] *Pseudo-Macro*

Expand to ON or OFF depending on whether the system operator is on or off duty.

Format

[!OPERATOR]

This pseudo-macro does not accept arguments. The operator makes his presence or absence known by appropriate control commands to EXEC.

The !OPERATOR pseudo-macro pertains to the presence of a system operator; it's set by the EXEC OPERATOR command. It has no relation to the CLI OPERATOR command.

Macroname Switches

None

Argument Switches

None

Example

!OPERATOR is most useful when you want to change macro behavior based on the presence of an operator.

Given a macro including:

```
[!EQUAL, [!OPERATOR],ON]
QBATCH FILE1 FILE2
[!ELSE]
WRITE TRY AGAIN LATER
[!END]
```

The CLI will queue up FILE1 and FILE2 if the operator is on duty; otherwise, it will output the message to try again later.

PATHNAME *Command*

Display a complete pathname starting at the root directory.

Format

PATHNAME filename

This command returns the full pathname beginning from the root to the specified file. You must have Execute access to the file whose pathname you specified in the command line.

Command Switches

/1=	IGNORE, WARNING, ERROR, ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE, WARNING, ERROR, ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) PATHNAME = ↓
:UDD:USER:BETA
)
```

Displays the pathname of =, the working directory.

```
) PATHNAME TEST ↓
:UDD:USER:BETA:TEST
```

Displays the pathname of TEST, a son file.

[!PATHNAME] *Pseudo-Macro*

Expand to a file's full pathname.

Format

[!PATHNAME pathname]

A full pathname starts at the root directory and ends with the specified filename.

Macroname Switches

None

Argument Switches

None

Example

```
) CREATE/LINK NIM [!PATHNAME NIM.PR] ↓
)
```

Creates a link entry named NIM containing a full pathname to the program NIM.PR.

PAUSE *Command*

Delay the CLI.

Format

PAUSE {seconds / seconds.milliseconds}

seconds is a number between 0 and 65535.

milliseconds is a number between 0 and 999.

This command pauses the CLI for the specified number of seconds.

Command Switches

/1=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) PAUSE 8.50 ↓
)
```

Delays the CLI for 8.5 seconds.

PED *Utility*

Display the process environment.

Format

XEQ PED

This command calls the Process Environment Display program, which displays runtime data on all system processes and terminals. It is a privileged utility, and is described further in the *How to Generate and Run AOS* (093-000217) and *How to Generate and Run AOS/VS* (093-000243).

PERFORMANCE *Command*

Display information about the CLI.

Format

PERFORMANCE

This command displays the following information about the CLI:

System Calls	The number of system calls this CLI has made since the last PERFORMANCE command and the number of system calls since this CLI started
Shared	The number of 2K-byte pages of shared memory
Unshared	The current number of pages of unshared memory, the maximum possible number of pages of unshared memory, and the greatest number of pages of unshared memory since this CLI started
Stack Faults	The number of stack faults since this CLI started; that is, the number of times this CLI has grown in 2K-byte memory pages

Command Switches

/1=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) PERFORMANCE ↓
153/489 SYSTEM CALLS
SHARED: 18 PAGES
UNSHARED: CURRENT 2 PAGES,
POSSIBLE 14 PAGES, HIGHEST 4 PAGES
5 STACK FAULTS
)
```

PERMANENCE *Command*

Set or display a file's permanence attribute.

Format

PERMANENCE pathname [*ON* / *OFF*]

A permanent file cannot be deleted unless you turn its permanence *OFF*. With the PERMANENCE command, you can protect key files from accidental deletion. You can use templates in the pathname argument. The PERMANENCE attribute only protects the file from specific deletion. The file's parent directory can be deleted even though it contains a permanent file. Also, you can overwrite a permanent file if you use LOAD_II (AOS/VS only) with the /DPERMANENT switch.

Command Switches

/1=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/V		Display the filename with its PERMANENCE attribute.

Argument Switches

None

Example

```
) PERMANENCE  ZONIS ↵
OFF
) PERMANENCE  ZONIS  ON ↵
) PERMANENCE/V  ZONIS ↵
ZONIS     ON
)
```

First, displays the PERMANENCE status of file ZONIS. Second, sets PERMANENCE for the file to ON. Last, displays and verifies the current PERMANENCE setting for the file.

[!PID] *Pseudo-Macro*

Expand to your CLI's process ID.

Format

[!PID]

This pseudo-macro does not accept arguments. This pseudo-macro always returns a three-digit number.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE MY PID IS [!PID] )
MY PID IS 017
)
```

PL1 *Language*

Compile a PL/I source file.

Format

XEQ PL1 source-pathname

or

PL1 source-pathname

PL1 is a macro that invokes the PL/I language. PL/I is a high-level language compatible with ANSI standard PL/I. For more on the PL1 language see the *PL/I Reference Manual (AOS/VS)* (093-000270), the *PL/I Reference Manual (AOS)* (093-000204), and the *Plain PL/I (A PL/I Primer)* (093-000216).

PL1LINK *Language*

Link object modules to form an executable PL/I program.

Format

PL1LINK main-objectmodule
[subprogram-objectmodule]...

PL1LINK is a macro that invokes the Link utility, to make PL/I object modules into an executable program. The PL1LINK macro will accept the switches of the Link utility. For a list of these switches, see the *AOS Link User's Manual* (093-000254) or the *AOS/VS Link and Library File Editor (LFE) User's Manual* (093-000245).

POP *Command*

Return to the previous environment level.

Format

POP

This command restores the previous environment's settings of CHARACTERISTICS, CLASS1, CLASS2, DATAFILE, DEFACL, DIRECTORY, LEVEL, LISTFILE, LOGFILE, PROMPT, SCREENEDIT, SEARCHLIST, STRING, SUPERPROCESS, SUPERUSER, SQUEEZE, TRACE, and VAR0 through VAR9. It preserves none of the current settings. If your current LEVEL is 0, the POP command causes a CLASS1 exceptional condition.

Command Switches

/1=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/V		Verify new level.

Argument Switches

None

Example

```
) LEVEL ↵
LEVEL 1
) POP/V ↵
LEVEL 0
)
```

First displays current level, then POP, to the previous environment and verifies the new level.

PREDITOR *Utility*

Create and edit user profiles.

Format

XEQ PREDITOR

You must have the Superuser privilege in order to use PREDITOR, though it does not need to be activated.

See the appropriate system manager's guide for more information about PREDITOR.

PREFIX *Command*

Set or display the prefix string.

Format

PREFIX *[argument]...*

The prefix string is the prompt the CLI displays to indicate its readiness to receive input. This command lets you display the current prefix string or specify up to 24 characters to be output at the prompt. The initial prefix is the right parenthesis. The other special characters are output in addition to the prefix: for example, & for line continuation, and * for Superuser.

Any character is valid as part of the prefix string. To enter certain special characters, you must use the !ASCII pseudo-macro, with the most significant bit set (that is, add 200 octal to the octal value of the character). These special characters are the control characters (octal values 0–37), parentheses (octal values 50 and 51), and angle brackets (octal values 74 and 76).

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/I		Set the prefix to the initial value, that is, a right parenthesis (no arguments allowed).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by **pathname** instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

PREFIX (continued)

Example

```
) PREFIX [!ASCII 207] HELLO ↵
HELLO PREFIX/I ↵
)
```

First, sets the prompt to ring a bell (the bell character is octal 7), and to write HELLO. The second command line shows the new prefix string. PREFIX with the /I switch resets the prefix string to its initial state, a right parenthesis.

PREVIOUS *Command*

Display the previous environment's settings.

Format

PREVIOUS

This command displays the previous level's environment parameter values. These parameters are CHARACTERISTICS, CLASS1, CLASS2, DATAFILE, DEFACL, DIRECTORY, LEVEL, LISTFILE, LOGFILE, PROMPT, SCREENEDIT, SEARCHLIST, SQUEEZE, STRING, SUPERPROCESS, SUPERUSER, TRACE, and VAR0–VAR9. (See Chapter 4 for more about CLI levels.) If you issue this command while in level 0, you will receive an error message.

Command Switches

/1=	IGNORE WARNING ERROR ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE WARNING ERROR ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) LEVEL ↓
LEVEL 1
) PREVIOUS ↓
LEVEL                0
SUPERUSER            OFF
SUPERPROCESS         OFF
SCREENEDIT           OFF
SQUEEZE              OFF
CLASS1               ABORT
CLASS2               WARNING
TRACE
VARIABLES            0 0 0 0 0
                     0 0 0 0 0
LISTFILE             @LIST
DATAFILE             @DATA
LOGFILE
DIRECTORY            :UDD:JOHN
SEARCHLIST           :PER,:UTIL,:
DEFACL               JOHN,OWARE
STRING
PROMPT
CHARACTERISTICS
                     /605X/LPP=24/CPL=80
                     /ON/ST/EB0/ULC/WRP
                     /OFF/SFF/EPI/8BT/SPO
                     /RAF/RAT/RAC/NAS OTT
                     /EOL/UCO/LT/FF/EB1/PM
                     /NRM/MOD /TO/TSP/PBN/ESC
                     /FKT/NNL
)
```

First displays the current level. Then, displays all the previous environment's settings on an AOS system. AOS/VS has more characteristics.

PRIORITY *Command*

Set or display the priority of the CLI or a subordinate process.

Format

PRIORITY [{ *username:procname* / *process-ID* } *new-priority*]

This command sets the priority of the CLI or a subordinate process. You cannot set the priority to a value higher than the CLI's priority unless you have the Superprocess privilege. If Superprocess is ON, you can change the priority of any process, not just a subordinate process.

new-priority is a decimal number greater than or equal to the priority at which the process was created. If you do not input a *new-priority*, the CLI displays the priority of the existing process. If you do specify a *new-priority*, the CLI will change the selected process's priority to the new value.

Command Switches

/1=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) PROCESS SMITH:SON1 ↓
PID 17
) PRIORITY 17 1 ↓
)
```

Sets SON1's priority to 1.

PROCESS *Command*

Create a process.

Format

PROCESS pathname *[argument-to-new-process]*

This command creates a son process with the program specified by pathname. You select the new process's type, priority, and privileges via command switches. Note that if you use the /IOC switch or the /STRING switch, you must also select /BLOCK.

The arguments to the new process are placed in the initital IPC message to the new process. The new process can access the arguments through the ?GTMES system call. See Appendix B for details.

The list file and data file passed to the son process (with the /LIST and /DATA switches) are the CLI's generic list file and data file. CLIs that are the son of EXEC and are running on terminals have no generic list and data files. CLIs that are the son of EXEC and are running in a batch stream have a generic list file created by EXEC and named username.LIST.sequence-number. They do not have a generic data file.

The CLI first tries to run pathname.PR. If that fails, it tries pathname.

If you omit all privilege switches, the new process has no privileges. If you use /DEFAULT, then the new process has the same privileges as the creating process.

NOTE: If you omit the /PREEMPTIBLE and /RESIDENT switches, the process is swappable.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/ACCESSDEVICES		Allow the new process to identify and access user devices via system calls ?IDEF, ?DEBL, and ?STMAP.
/BLOCK		Block this CLI until the new process terminates. If you omit this switch, the CLI does not block and displays the new process's ID. The CHECKTERMS command can display the termination message from the created process when it terminates. You must use /BLOCK if you use the /IOC or /STRING switches.
/BREAK		(AOS/VS only) Create a break file if an error trap or fatal termination occurs; the default is no break file.
/BSON		Block the son process until explicitly unblocked.
/CALLS=number		Maximum number of concurrent system calls for the new process; default is the same as for the creating process.
/CHLOGICALTYPE		(AOS/VS only) Allow the new process to change its logical type (32-bit or 16-bit).
/CHPRIORITY		Allow the new process to change its priority.
/CHTYPE		Allow the new process to create any other type of process. Also permit the new process to change its own process type.
/CHUSERNAME		Allow the new process to create a process with a different username than its own.
/CHWSS		(AOS/VS only) Allow the new process to change its working set size.
/CONSOLE		Make the new process's console the same as the creating process's; The default is no console.
/CONSOLE =consolename		Make consolename the new process's console.

/CPU=s	Limit the CPU time for a new process, where s is a number of seconds between 0 and 4,294,967.
/DACL	Do not pass the default ACL to the new process.
/DATA	Make the new process's generic @DATA filename the same as the creating process's; default is no @DATA.
/DATA=pathname	Make pathname the new process's generic @DATA filename.
/DEBUG	Start the new process in the debugger.
/DEFAULT	Give the new process the same privileges as the creating process.
/DIRECTORY	Make the new process's initial directory the creating process's initial directory; default is the creating process's working directory.
/DIRECTORY=pathname	Make pathname the new process's initial directory.
/DUMP	Append a system dump to the breakfile statistics (AOS/VS only).
/INPUT	Make the new process's generic @INPUT filename the same as the creating process's; default is no @INPUT.
/INPUT=pathname	Make pathname the new process's generic @INPUT filename.
/IOC	Make the new process's @INPUT, @OUTPUT, and @CONSOLE generic filenames the same as the creating process's. You must use /BLOCK if you use the /IOC switch.
/IOC=consolename	Make consolename the new process's generic @INPUT, @OUTPUT, and @CONSOLE names.
/IPCUSAGE	Allow the new process to issue the primitive IPC calls.
/L	Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname	Write CLI output to the file specified by pathname instead of to @OUTPUT.
/LIST	Make the new process's generic @LIST filename the same as the creating process's. The default is no @LIST.
/LIST=pathname	pathname is the new process's generic @LIST.
/MEMORY=pages	Make pages the maximum memory size of the new process in 2K-byte pages. The default for AOS is the same as the creating process's. For AOS/VS, the default is minimum of top of shared or 512 Mbytes. For 16-bit programs running under AOS/VS, the default is minimum of top of shared or 64 Kbytes.
/NAME=name	Make name the simple process name for the new process. (If you omit this switch, the system assigns the name.)
/NOBLOCKPROC	Allow the new process to create another process without blocking.
/OUTPUT	Make the new process's generic @OUTPUT filename the same as the creating process's; the default is no @OUTPUT.
/OUTPUT=pathname	Make pathname the new process's generic @OUTPUT filename.
/PMGRPRIVILEGES	Allow the new process all the rights of the peripheral manager.

PROCESS (continued)

/PRIORITY=number	Make number the new process's priority; the default is the same as creating process's priority
/PREEMPTIBLE	Make the new process preemptible.
/Q	Set SQUEEZE to ON for this command.
/RESIDENT	Make the new process resident.
/SONS	A son process may create the same number of processes as the creating process, minus one; default is zero. If you omit /SONS, /SONS=number or /UNLIMITEDSONS, the default for creating sons is zero.
/SONS=number	Make number the maximum number of son processes that the new process can create. If you omit /SONS, /SONS=number or /UNLIMITEDSONS, the default for creating sons is zero.
/STRING	Store the program termination IPC message in the current STRING instead of displaying it. You must use /BLOCK with this switch.
/SUPERPROCESS	Allow the son process to enter Superprocess mode.
/SUPERUSER	Allow the son process to enter Superuser mode.
/UNLIMITEDSONS	Allow the new process the option of creating an unlimited number of son processes. If you omit /SONS, /SONS=number, or /UNLIMITEDSONS, the default is zero.
/USERNAME=name	Make name the new process's username; the default ACL is username OWARE.
/WSMAX=pagenum	(AOS/VS only) Specify maximum number of pages allowed in main memory at one time; the default is dynamically set by the system.
/WSMIN=pagenum	(AOS/VS only) Specify minimum number of pages that must be in main memory; the default is dynamically set by the system.

Argument Switches

Use any argument switches appropriate for the program specified in pathname.

Example

```
) PROCESS/IOC=@CON1 UPDATE ↵
PID: 13
)
```

Creates a swappable son process with @INPUT, @OUTPUT, and @CONSOLE equivalent to @CON1 and with UPDATE as its program. The CLI displays the process ID of the subordinate process (13).

```
) PROCESS/BLOCK/IOC LAMBDA 1 ↵
)
```

Creates a swappable son process with the same generic @INPUT, @OUTPUT, and @CONSOLE as this CLI, and blocks the CLI until this son terminates. The new process's program is LAMBDA, and it has access to the argument 1 via the ?GTMES system call (see Appendix B for a description of ?GTMES).

PROMPT *Command*

Set or display the current prompt setting.

Format

PROMPT *[command]...*

This command allows you to display the prompt or to specify up to eight CLI commands that the system will execute before it issues the prompt. When setting a PROMPT argument, you must enter only the CLI command name (no associated arguments or switches — see Example). You can use only those commands that do not require an argument. One difference between the PROMPT command and the PREFIX command is that the PROMPT command executes other CLI commands while the PREFIX command only displays whatever is the prefix string.

Command Switches

/1=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS1 to the specified severity level for this command.
/2=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS2 to the specified severity level for this command.
/K		Set PROMPT to null (no arguments allowed).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set PROMPT to previous environment's PROMPT (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) PROMPT TIME DATE DIRECTORY ↵
9:32:16
26-NOV-84
:UDD:USER:PHIL
) PROMPT ↵
TIME
DATE
DIRECTORY
9:32:18
26-NOV-84
:UDD:USER:PHIL
) PROMPT/K ↵
)
```

First, sets PROMPT to the CLI commands that you want the system to execute before it issues the prompt character. (Note that there are no optional arguments or switches.) Then, displays PROMPT; then sets the PROMPT to null.

PRTYPE *Command*

Set or display the type of an inferior process.

Format

PRTYPE [{ *username:procname* / *process-ID* } [*PREEMPTIBLE* / *RESIDENT* / *SWAPPABLE*]]

You can't change your process type unless you have the privilege ?PVTY, or Superprocess mode is ON. If Superprocess mode is ON, you may change the type of any process, not just an inferior process. The types are:

RESIDENT	Process cannot be swapped
REEMPTIBLE	Process can be swapped only after all swappable processes have been blocked
SWAPPABLE	Process can be swapped, as needed (this is the default for all processes)

Command Switches

/1= (IGNORE / WARNING / ERROR / ABORT)	Set CLASS1 to the specified severity level for this command.
/2= (IGNORE / WARNING / ERROR / ABORT)	Set CLASS2 to the specified severity level for this command.
/L	Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname	Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q	Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) PROCESS/PREEMPTIBLE SMITH:PROGA ↵
PID: 14
) PRTYPE 14 SWAPPABLE ↵
)
```

First, creates a preemptible son process with program PROGA. (The system assigns the new process a PID of 14.) Then, sets process 14's type to swappable.

PUSH *Command*

Descend to a new environment.

Format

PUSH

This command saves the current environment then pushes a level. You may now change the environment settings CHARACTERISTICS, CLASS1, CLASS2, DATAFILE, DEFACL, DIRECTORY, LEVEL, LISTFILE, LOGFILE, PROMPT, SCREENEDIT, SEARCHLIST, STRING, SQUEEZE, SUPERPROCESS, SUPERUSER, TRACE, and VAR0 through VAR9, using the appropriate CLI commands. See Chapter 4 for a description of the CLI environment.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/V		Display the new environment's level.

Argument Switches

None

Example

```
) LEVEL ↵
LEVEL 0
) PUSH/V ↵
LEVEL 1
)
```

First, displays the current environment's level; then Pushes a level (thereby saving the current environment) and displays the new level setting.

QBATCH *Command*

Create and submit a batch job file.

Format

QBATCH argument *[argument]...*

When you type this command, the CLI creates a batch job file in your working directory and places an entry for it on the batch queue. The batch job file begins with commands that set the batch job's working directory, search list, and default ACL to their current setting. If you issue the QBATCH command without /I or /M, the remainder of the command line becomes the batch job. The EXEC utility deletes the job file after the job runs.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/AFTER=date:time		date:time is in the form dd-mmm-yy:hh:mm:ss. Process this request after date and time. Note that the /AFTER switch effectively guarantees that the request will not be processed before a certain time. The request will remain in the queue while the /AFTER switch is in effect and will gain priority by virtue of its age. You can use a plus sign (+) to specify a relative time for process delay. For example, /AFTER=+12 says don't process until at least 12 hours have passed.

QBATCH (continued)

/CPU=time — Limit CPU time for batch jobs, where time specifies the maximum amount of CPU time that the request can use. This switch accepts time in the form hh:mm:ss, where minutes and seconds are optional. You must allow enough time for all processes created in the batch job. If you omit this switch, EXEC will assume 1 minute of CPU time.

This switch is acted on only if the operator has set a time limit for jobs in the stream; otherwise, the switch is ignored. If the limit is on, and the time specified by the switch exceeds the limit, EXEC rejects the command.

/DESTINATION=string — Print string in block letters at the top of any header or trailer pages. The default destination string is username.

/HOLD — Hold the entry until you explicitly unhold it with the QUNHOLD command.

/I — Take the contents of the file from subsequent lines of the @INPUT file. You must terminate the input with a single right parenthesis), and a NEW LINE (no arguments allowed).

/JOBNAME=name — Name the entry name. You can use name to QHOLD, QUNHOLD, or QCANCEL the job. The jobname must contain at least one alphabetic character. The default jobname is null.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/M — Take the contents of the file from subsequent lines of the current macro body. The last line of the macro file must contain a single right parenthesis). (No arguments allowed).

/NORESTART — If the system fails while it is processing this entry do not restart the job.

/NOTIFY — Cause EXEC to send a message back to your terminal when the queue request is completed.

/OPERATOR — Do not run this job if no operator is present. You should use this switch when submitting a batch job containing a MOUNT request.

/Q — Set SQUEEZE to ON for this command.

/QLIST=pathname — Set the generic list file of the batch process to pathname. pathname may not be a queuename.

/QOUTPUT=pathname — Set the generic output file of the batch process to pathname. pathname should not be a queuename.

/QPRIORITY=n — Give this job priority n ($0 < n < 255$). n job priority specified in your user profile.

/S — Store the sequence number in STRING where you can use it as an argument to commands via the !STRING pseudo-macro.

/V — Display the name of the batch job file.

Argument Switches

Use any argument switches appropriate for batch job specified in argument.

Example

```
) QBATCH XEQ MASM FILE3 ⤸
QUEUED, SEQ=65, QPRI=128
)

) QBATCH/I ⤸
))XEQ MASM FILE3 ⤸
))XEQ BIND FILE3 ⤸
))XEQ FILE3 ⤸
))) ⤸
QUEUED, SEQ=66, QPRI=128
)

) QBATCH/V XEQ MASM FILE3 ⤸
:UDD:CLJ: ?009.CLI.001.JOB
QUEUED, SEQ=67, QPRI=128
)
```

The 009 indicates the process ID of the issuing CLI. The 001 is included to make the filename unique; i.e., the next batch command you execute may generate file ?009.CLI.002.JOB.

QCANCEL *Command*

Cancel a queue entry.

Format

QCANCEL { seq-no / jobname } [*seq-no* / *jobname*] ...

This command removes the specified entry from the queue to which it was submitted. Use the QDISPLAY command to find the seq-no that EXEC assigned to your entry. If your user process runs under EXEC, you can cancel only your own entries in the queue. You can cancel all jobs in a queue with a given jobname with one command by specifying the jobname. If the jobname is null, enter two commas as the argument. This will cancel all jobs with your username and a null jobname.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

QCANCEL (continued)

Argument Switches

None

Example

```
)QCANCEL JOB1 ↓
)
```

Removes the entry for JOB1 from the BATCH_INPUT queue.

```
)QCANCEL ,, ↓
)
```

This cancels all jobs which you queued up that have a null jobname.

QDISPLAY *Command*

Display queue information.

Format

QDISPLAY *[hostname]*

This command displays the name and type of each queue maintained by the operating system. If you are allowed to place entries in a queue, this command will display the word OPEN with the queue name and type. If you do not provide an argument, QDISPLAY will display information from the local queues. Otherwise, the QDISPLAY will display information about the specified remote queues.

If you omit switches, QDISPLAY lists all queue names and their entries. Entries preceded by an asterisk are currently being processed; other entries are preceded by a letter that indicates their status.

Status Letters

*	Active job
A	Unexpired /AFTER switch in effect
B	/BINARY switch in effect
C	Canceled by user (QCANCEL command)
D	User /DELETE switch in effect
E	Held by operator
F	Canceled by operator
G	User /NORESTART switch in effect.
H	Held by user
N	/NOTIFY switch in effect
O	/OPERATOR switch in effect
R	Restarted
S	Queued by SUPERUSER

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.

/L	Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname	Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q	Set SQUEEZE to ON for this command.
/QUEUE=queuename	Display only the queuename. This switch may appear more than once in a command. Permanent queuenames are: BATCH_INPUT BATCH_OUTPUT BATCH_LIST Check with your operator for local queuenames.
/SUMMARY	List only queuenames, types, and entry summaries.
/TYPE=type	Display queues of type only. This switch may appear more than once. Queue types are: BATCH PRINT PUNCH (AOS only) PLOT HAMLET RJE80 FTA
/V	Display appropriate column headings and more complete information for each queue that has entries and that is to be displayed. This switch has no effect if /SUMMARY is also present.

Argument Switches

None

Example

) QDISPLAY ↵

Displays information about all queues.

) QDISPLAY/TYPE=BATCH/TYPE=PRINT ↵

Displays information about the batch and print queues.

) QDISPLAY/QUEUE=BATCH_INPUT ↵

Displays information about the queue named BATCH_INPUT.

QHOLD *Command*

Hold a queue entry.

Format

QHOLD { seq-no / jobname } [*seq-no* / *jobname*] ...

This command allows you to temporarily suspend a queue entry. You can unhold the entry with the QUNHOLD command. With QHOLD, you can hold only your own entries and those entries which are not active. After a job becomes active, you must use the QCANCEL command.

To hold all jobs in a queue with the same jobname, specify the jobname. If the jobname is null, then enter two consecutive commas as the argument. This will hold all jobs with your username and a null jobname.

Command Switches

/1= { IGNORE / WARNING / ERROR / ABORT } — Set CLASS1 to the specified severity level for this command.

/2= { IGNORE / WARNING / ERROR / ABORT } — Set CLASS2 to the specified severity level for this command.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/Q — Set SQUEEZE to ON for this command.

Example

```
) QSUBMIT/QUEUE=BATCH_INPUT& ↵
&)/JOBNAME=JOVIAL  FILE1 ↵
.
.
.
) QHOLD  JOVIAL ↵
)
```

Holds jobname JOVIAL until a subsequent QUNHOLD command releases it.

QPLOT *Command*

Place an entry on a plotter queue.

Format

QPLOT pathname *[pathname]...*

This command places an entry on a digital plotter queue. Note that the command does not plot the file, but merely queues it to the plotter; so don't delete or modify the file until the system outputs it. The system always plots data exactly as it appears in the file. EXEC does not record billing parameters.

You may use templates in the pathname argument(s).

Command Switches

/1= { IGNORE / WARNING / ERROR / ABORT } — Set CLASS1 to the specified severity level for this command.

/2= { IGNORE / WARNING / ERROR / ABORT } — Set CLASS2 to the specified severity level for this command.

/AFTER=date:time — Process this request after date and time. date:time is in the form dd-mmm-yy:hh:mm:ss. Note that the /AFTER switch effectively guarantees that the request will not be processed before a certain time. The request will remain in the queue while the /AFTER switch is in effect and will gain priority by virtue of its age. You may use a plus sign (+) to specify a relative time for process delay. For example,

/AFTER=+12

says don't process until at least 12 hours have passed.

/COPIES=n — Produce n copies of the file. 1< n <25. The default is n= 1.

/DELETE — Delete the pathnames after plotting them.

/FORMS=type	Specify that special forms **type** must be used. Check with your operator for local forms types. If you omit this switch, the system will use the standard forms.
/HOLD	Hold the entry until you explicitly unhold it with the QUNHOLD command.
/L	Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname	Write CLI output to the file specified by **pathname** instead of to @OUTPUT.
/NORESTART	Do not restart the plotting if the system fails while it is plotting this file.
/NOTIFY	Cause EXEC to send a message back to your terminal upon completion of the queue request.
/OPERATOR	Do not run this job if no operator is present. You should use this switch when submitting a job that would require special forms.
/Q	Set SQUEEZE to ON for this command.
/QPRIORITY=n	Give the entry priority **n**. 1 is the highest priority and 255 is the lowest priority. **n** cannot be less than the priority specified in your user profile (m). If you omit this switch, the system calculates the priority as follows: n=(m+255)/2
/QUEUE=queuename	Submit job to **queuename** instead of to the default queue. The queue type must be PLOT.
/S	Store the sequence number in STRING where you can use it as an argument to commands via the !STRING pseudo-macro.
/V	Display the names of the queued files.

Argument Switches

None

Example

```
) QPLOT FILE1 FILE2 ↵
QUEUED, SEQ=651, QPRI=127
QUEUED, SEQ=652, QPRI=127
)
```

Queues FILE1 and FILE2 to a digital plotter output queue.

```
) QPLOT/DELETE FILE3 ↵
QUEUED, SEQ=653, QPRI=127
)
```

Plots FILE3 then delete FILE3 when it is complete.

```
) QPLOT/COPIES=3/TITLES FILE4 ↵
QUEUED, SEQ=654, QPRI=127
)
```

Plots three copies of FILE4 and produce titles on each page.

QPRINT *Command*

Place an entry on the PRINT queue.

Format

QPRINT pathname *[pathname]...*

This command queues the file named in pathname to the line printer output queue. Note that this command does not print the file, but merely queues it to the printer; so don't delete or modify the file until the system outputs it.

You may use templates in the pathname argument(s).

Command Switches

/1= {IGNORE, WARNING, ERROR, ABORT} Set CLASS1 to the specified severity level for this command.

/2= {IGNORE, WARNING, ERROR, ABORT} Set CLASS2 to the specified severity level for this command.

/AFTER=date:time Process this request after date and time. date:time is in the form dd-mmm-yy:hh:mm:ss. Note that the /AFTER switch effectively guarantees that the request will not be processed before a certain time. The request will remain in the queue while the /AFTER switch is in effect and will gain priority by virtue of its age. You may use a plus sign (+) to specify a relative time for process delay. For example,

/AFTER=+12

says don't process until at least 12 hours have passed.

/BEGIN=n Start printing the file at page n. The default is n=1.

/BINARY Print in binary mode. This switch is valid only for devices that have binary mode enabled. Check with your operator for local binary devices.

/COPIES=n Produce n copies of the file. The default is n=1.

/DELETE Delete files after printing them.

/DESTINATION=string Print string in block letters at the top of any header or trailer pages. The default destination string is username.

/END=n Stop printing the file at page n. The default is n=last page.

/FOLDLONGLINES Do not truncate long lines. Continue them on next line of the listing.

/FORMS=type Print on special forms type. Check with your operator for local forms types. If you omit this switch, the system will use standard forms.

/HOLD Hold the entry until you explicitly unhold it with the QUNHOLD command.

/L Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname Write CLI output to the file specified by pathname instead of to @OUTPUT.

/NORESTART Do not restart the listing if the system fails while it is printing this file.

/NOTIFY Tells EXEC to send a message back to your terminal upon completion of the queue request.

/OPERATOR Do not run this job unless an operator is present. You should use this switch when submitting a job which will require special forms.

/PAGES=n	Your operator will tell you whether or not to specify /PAGES. If you do, specify n as the maximum number of pages that you will print. If you omit this switch, EXEC estimates the page limit as follows: **pages = (bytes-in-file)/1000 + 4** If the operator-set page limit is on, and the value specified by this switch exceeds the limit, then EXEC rejects the command.
/Q	Set SQUEEZE to ON for this command.
/QPRIORITY=n	Give the entry priority n. 1 is the highest priority and 255 is the lowest priority. n cannot be less than the priority specified in your user profile (m). If you omit this switch, the system calculates the priority as follows: **n = (m+255)/2 .**
/QUEUE=queuename	Submit the job to **queuename** instead of to the default queue. The queue type must be PRINT.
/S	Store the sequence number in STRING where you can use it as an argument to commands via the !STRING pseudo-macro.
/TITLES	Print each page with a title line, consisting of pathname, date and time last modified, and page number. By default, the system prints no titles.
/V	Display the names of the queued files.

Argument Switches

None

Example

```
) QPRINT FILE1 FILE2 )
QUEUED, SEQ=655, QPRI=127
QUEUED, SEQ=656, QPRI=127
)
```

Queues FILE1 and FILE2 to the line printer output queue.

```
) QPRINT/DELETE/FOLDLONGLINES FILE3 )
QUEUED, SEQ=657, QPRI=127
)
```

Prints FILE3, folding long lines. Deletes FILE3 when it finishes printing.

```
) QPRINT/COPIES=3/PAGES=75/TITLES FILE4 )
QUEUED, SEQ=658, QPRI=127
)
```

Prints three copies of FILE4 and produces titles on each listing page. Each listing begins on page 1. The total number of pages to print will not exceed 75.

QPUNCH *Command*

Place an entry on the punch queue (AOS only).

Format

QPUNCH pathname *[pathname]...*

This command places the file named in **pathname** on a paper tape punch queue. Note that this command does not punch the file, but merely queues it to the punch; so don't delete or modify the file until the system outputs it.

You may use templates in the pathname argument(s).

Command Switches

/1= {IGNORE / WARNING / ERROR / ABORT}
Set CLASS1 to the specified severity level for this command.

/2= {IGNORE / WARNING / ERROR / ABORT}
Set CLASS2 to the specified severity level for this command.

/AFTER=date:time
Process this request after dd-mmm-yy:hh:mm:ss. **date:time** is in the form **dd-mmm-yy:hh:mm:ss.** Note that the /AFTER switch effectively guarantees that the request will not be processed before a certain time. The request will remain in the queue while the /AFTER switch is in effect and will gain priority by virtue of its age. You may use a plus sign (+) to specify a relative time for process delay. For example,

/AFTER=+12

says don't process until at least 12 hours have passed.

/COPIES=n
Produce n copies of the file. $1 < n < 25$. The default is n=1.

/DELETE
Delete files after punching them.

/FEET=n
n is the maximum number of feet of tape that you will punch. If you omit this switch, the system estimates the limit by the size of the file. If you specify /COPIES, use this switch.

/FORMS=type
Specify that special forms **type** must be used. Check with your operator for local forms types. If you omit this switch, the system will use the standard forms.

/HOLD
Hold the entry until you explicitly unhold it with the QUNHOLD command.

/L
Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname
Write CLI output to the file specified by **pathname** instead of to @OUTPUT.

/NORESTART
Do not restart the listing if the system fails while it is punching this file.

/NOTIFY
Tell EXEC to send a message back to your terminal upon completion of the queue request.

/OPERATOR
Do not run this job unless an operator is present. You should use this switch when submitting a job which will require special tape.

/Q
Set SQUEEZE to ON for this command.

/QPRIORITY=n
Give the entry priority n. 1 is the highest priority and 255 is the lowest priority. n cannot be less than the priority specified in your user profile (m).

If you omit this switch, the system calculates the priority as follows:

n = (m+255)/2

/QUEUE=queuename	Submit the job to queuename instead of to the default queue. The queue type must be PUNCH.
/S	Store the sequence number in STRING where you can use it as an argument to commands via the !STRING pseudo-macro.
/V	Display the names of the queued files.

Argument Switches

None

Example

```
) QPUNCH FILE1 FILE2 ↓
QUEUED, SEQ=659, QPRI=127
QUEUED, SEQ=660, QPRI=127
)
```

Queues FILE1 and FILE2 to the paper tape punch output queue.

```
) QPUNCH/DELETE FILE3 ↓
QUEUED, SEQ=661, QPRI=127
)
```

Queues FILE3, then deletes FILE3 when complete.

```
) QPUNCH/COPIES=3/FEET=75 FILE4 ↓
QUEUED, SEQ=662, QPRI=127
)
```

Queues three copies of FILE4. Will not punch more than 75 feet of paper tape.

QSNA *Command*

Place an entry on the Systems Network Architecture (SNA) queue.

Format

QSNA pathname *[pathname]...*

Submits a request on the System Networking Architecture (SNA) queue for batch type data transfer over a Data General SNA network to an IBM Remote Job Entry (RJE) system. The pathname is the file you want to send to the host. If you do not use the /QOUTPUT= switch, SNA/Remote Job Entry (RJE) creates a generic output file for the QSNA Command in the form

:UDD:username:RJE.OUTPUT.n.

n is the job sequence number the system assigns the job when you execute the command. For more information about SNA/RJE, see the *SNA/RJE Operator's and User's Guide.*

You can use templates in the pathname argument and you can abbreviate switch names.

Command Switches

/1= {IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2= {IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/AFTER=date:time	Execute command on or after specified date and/or time. The request remains in the SNA queue while the /AFTER switch is in effect. You can use a plus sign (+) to specify a relative time for process delay. For example, QSNA/AFTER=+3 means, "Transfer the file as soon as possible, 3 hours from now." For the date:time format: to specify the date only, use dd-mmm-yy; to specify the time only, use hh:mm:ss. If you omit the time, the system assumes midnight.

QSNA (continued)

/CONCATENATE	Concatenate the data from the sources listed in the pathname arguments into one data stream and transfer them.
/DELETE	Delete the source file(s) after completing the transfer. Use this switch only if the filename is not a devicename.
/ERCL=n	Set exchange record length (n) for transfer. (n is a number from 0 to 255).
/HOLD	Holds the entry on the queue until you explicitly release it with the QUNHOLD command.
/L	Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname	Write CLI output to the file specified by pathname instead of to @OUTPUT.
/MEDIUM=	Specify the device class for the device or file whose data you are submitting to the SNA queue. Device choices are: CARD, CONSOLE, or EXCHANGE. The default device is CARD.
/NORESTART	Do not restart (start at the beginning of the file) the transfer if the system fails during the transfer. The default is to restart processing.
/NOTIFY	Display a message on your terminal screen when your request is completed.
/OPERATOR	Start processing this job only if a system operator is on duty.
/Q	Set SQUEEZE to ON for this command.
/QOUTPUT=pathname	Set the output file of the batch process to the specified pathname.
/QPRIORITY=n	Establish a priority for the request, where n is a number between 1 and 255. (1 is the highest priority; 255 is the lowest.) You cannot request a priority higher than that allowed in your user profile.
/QUEUE=queuename	Submit the job to queuename instead of to the default queue.
/STREAM=n	Specify the SNA/RJE stream number which will process the request. (n is in the range of 0 to 6; the default is 1. 0 lets SNA/RJE choose any stream).
/SUBADDRESS=n	Set the subaddress to a number equal to the subaddress of the device or file whose data you are submitting to the SNA queue. (n is a number from 0 to 15; the default is 0).
/TRANSPARENT	Send the data to the IBM host as transparent data, which means that the host does not interpret the data.
/TYPE=	Direct SNA/RJE to read data from the device or file as specified by type. Type options are: LINE—Performs data-sensitive read (default). Translates ASCII data to EBCDIC. The default record length is 80 bytes. SEQUENTIAL—Performs dynamic read of 80. characters; translates ASCII data to EBCDIC. The default record length is 80 bytes. EBCDIC—Performs dynamic read of the number of characters specified when SNA/RJE was configured; does not translate data. The default record length is 80 bytes.

HOLLERITH—Translates data from Hollerith to EBCDIC. Performs dynamic reads. (Valid for card readers only).

/V — Verify the transaction by displaying the name(s) pathname(s) of the queued file.

Example

```
) QSNA/MED=E/ERCL=100/TYPE=S  JOB1 )
)
```

Queues file JOB1 on the SNA queue. Transfers data in JOB1 to a remote IBM host. JOB1 contains a job for the host to execute. Sets the medium to exchange, reads the file with an exchange record length of 100 characters using dynamic reads.

QSUBMIT *Command*

Place an entry on a batch or spool queue.

Format

QSUBMIT pathname *[pathname]...*

This command queues an entry to the specified queue for each pathname you supply in the argument list. If you omit the /QUEUE= switch, QSUBMIT assumes the batch input queue because you normally use the CLI commands QPLOT, QPRINT, or command switches for the other queues.

Do not use QSUBMIT to submit batch jobs in stacked format. Stacked format commonly applies to jobs submitted on punched cards. This procedure is described in Appendix D.

You may use templates in the pathname argument(s).

Command Switches

/1= { IGNORE | WARNING | ERROR | ABORT } — Set CLASS1 to the specified severity level for this command.

/2= { IGNORE | WARNING | ERROR | ABORT } — Set CLASS2 to the specified severity level for this command.

/AFTER=date:time — Process this request after date and time. date:time is in the form dd-mmm-yy:hh:mm:ss. Note that the /AFTER switch effectively guarantees that the request will not be processed before a certain time. The request will remain in the queue while the /AFTER switch is in effect and will gain priority by virtue of its age. You may use a plus sign (+) to specify a relative time for process delay. For example,

/AFTER=+12

says don't process until at least 12 hours have passed.

QSUBMIT (continued)

/BINARY — Print in binary mode. This switch is valid only for devices with binary mode enabled. Check with your operator for local binary devices.

/CPU=time — Limit CPU time for batch jobs, where time specifies the maximum amount of CPU time that the request can use. This switch accepts time in the form hh:mm:ss, where minutes and seconds are optional. You must allow enough time for all processes created in the batch job. If you omit this switch, the system will assume one minute of CPU time.

This switch is acted on only if the operator has set a time limit for jobs in the stream; otherwise, the switch is ignored. If the limit is on, and the time specified by the switch exceeds the limit, EXEC rejects the command.

/DELETE — Delete the pathname(s) after processing.

/DESTINATION=string — Print string in block letters at the top of any header or trailer pages. The default destination string is username.

/HOLD — Hold the entry until you explicitly unhold it with the QUNHOLD command.

/JOBNAME=name — Batch queues only. Name the entry name. Then you can use name to QHOLD, QUNHOLD, or QCANCEL the job. (The jobname must contain at least one alphabetic character.) The default jobname is null.

/L — Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname — Write CLI output to the file specified by pathname instead of to @OUTPUT.

/NORESTART — If the system fails while it is processing this entry, do not restart the job.

/NOTIFY — Tell EXEC to send a message back to your terminal upon completion of the queue request.

/OPERATOR — Do not run this job if no operator is present. Batch jobs requiring the MOUNT feature should be submitted with this switch.

/Q — Set SQUEEZE to ON for this command.

/QLIST=pathname — Set the generic list file of the batch process to pathname.

/QOUTPUT=pathname — Set the generic output file of the batch process to pathname. pathname should not be a queuename.

/QPRIORITY=n — Give the job priority n ($1 < n < 255$). n cannot be less than the priority specified in your user profile (m).

If you omit this switch, the system calculates the priority as follows:

$n = (m+255)/2$.

/QUEUE=queuename — Submit job to the queuename. If you omit this switch, QSUBMIT assumes BATCH_INPUT.

/S — Store the sequence number in STRING where you can use it as an argument to commands via the !STRING pseudo-macro.

/V — Display the names of the queued files.

/XW0=n — Place the value n in the ?XXW0 word of the ?EXEC system call packet. n must be a decimal number.

/XW1=n	Place the value n in the ?XXW1 word of the ?EXEC system call packet. n must be a decimal number.
/XW2=n	Place the value n in the ?XXW2 word of the ?EXEC system call packet. n must be a decimal number.
/XW3=n	Place the value n in the ?XXW3 word of the ?EXEC system call packet. n must be a decimal number.

Agument Switches

None

Example

```
) QSUBMIT FILE1 FILE2 )
QUEUED, SEQ=663, QPRI=127
QUEUED, SEQ=664, QPRI=127
)
```

Submits FILE1 and FILE2 to the BATCH_INPUT

```
) QSUBMIT/NORESTART/HOLD FILE3 )
QUEUED, SEQ=665, QPRI=127
)
```

Submits FILE3 to the BATCH_INPUT queue. Holds it until a subsequent QUNHOLD command releases it. Does not restart job FILE3 if the system fails while it is processing the file. The system calculates the job's priority as described above.

```
) QSUBMIT/AFTER=12:30 BATCHJOB )
QUEUED, SEQ=667, QPRI=127
)
```

The system will not process this job before 12:30 p.m.

QUNHOLD *Command*

Free a held queue entry.

Format

QUNHOLD { seq-no / jobname } [*seq-no* / *jobname*] ...

This command negates a previous QHOLD command on a queue entry. Note that you cannot QUNHOLD an entry that the operator has on hold. If you QHOLD a BATCH_INPUT entry by jobname, then you must QUNHOLD it by jobname. However, you can QUNHOLD any BATCH_INPUT entry by sequence-number.

You can unhold all jobs in a queue with a null jobname by entering two consecutive commas as the argument to the QUNHOLD command. This will unhold all jobs with your username and a null jobname.

Command Switches

/1= { IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2= { IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/L	Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname	Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q	Set SQUEEZE to ON for this command.

QUNHOLD (continued)

Argument Switches

None

Example

```
) QHOLD MYJOB ↓
.
.
.
) QUNHOLD MYJOB ↓
)
```

First holds, and then unholds, the batched job with the jobname MYJOB.

RDOS *Utility*

Read or write an RDOS dump file or disk.

Format

XEQ RDOS command *[argument] ...*

RDOS is the Real-time Disk Operating System. The RDOS utility reads an RDOS dump file or disk to, or writes it from, your AOS or AOS/VS working directory. You may use AOS or AOS/VS templates, but not RDOS templates, in any of the following five RDOS utility commands: LOAD, DUMP, GET, PUT, and LIST.

The LOAD command loads an RDOS dump file on 9-track magnetic tape into your working directory. The DUMP command dumps files from your working directory onto a 9-track magnetic tape in RDOS dump format. The GET, PUT, and LIST commands assume that an RDOS disk is on your system. In order to execute these commands, you must have previously specified the disk drive as part of your system during AOSGEN or VSGEN. Do not try to initialize the RDOS disk.

Restrictions

- The RDOS utility will not dump to, or load from, the following devices: paper tapes, cassettes, 7-track magnetic tapes.
- The RDOS utility will not GET, PUT, or LIST disks that AOS or AOS/VS do not support (e.g., 4047A, 4047B, 4237, 4238, 4048A, and 4057A).
- The RDOS utility will not load any files onto the RDOS disk (the LOAD command only loads tapes onto the AOS or AOS/VS disk).
- The RDOS utility will not dump any files from the RDOS disk (the DUMP command only dumps files from the AOS or AOS/VS disk).
- The RDOS utility will not XFER (in RDOS CLI terminology) or COPY (in AOS and AOS/VS CLI terminology) files; it only dumps and loads.

Commands

The DUMP, GET, LIST, LOAD, and PUT commands that follow are particular to the RDOS utility.

DUMP

Dump one or more user data or program files to an RDOS dump file. Attempts to dump a link file will result in dumping the resolution file, if it exists. You cannot dump peripherals, IPC files, queue files, or generic files.

Format

XEQ RDOS DUMP rdos-dumpfile
[aos-filename]...[templates]...

DUMP Switches

None

NOTE: The RDOS utility does not add the S attribute to the resulting RDOS file if the AOS file type is PRG or if the AOS filename ends in .PR or .SV. You will have to use the CHARACTERISTICS CLI command on your RDOS system to give the file an S attribute.

The RDOS utility will not make overlay files contiguous. You will have to use the local /C switch on the XFER command after you have loaded the dump file.

RDOS Switches

/A	Abort on an ABORT condition.
/L	List files on LISTFILE.
/L=pathname	List files on file specified by pathname.
/V	Verify each file transferred.

Argument Switches

/C	Convert NEW LINEs to carriage returns.
/N	Do not transfer files matching this template.

GET

Get one or more files from an RDOS disk and place them in a subdirectory of your working directory. You must create the subdirectory before you issue this command.

Format

@XEQ RDOS GET/disk=rdos-diskunit
[rdos-filename]...[templates]...

GET Switches

/A	Abort on an ABORT condition.
/L	List files on LISTFILE.
/L=pathname	List files on file specified by pathname.
/N	Do not load, just verify filenames.
/T	Move the contents of all RDOS directories designated in the command line. Without this switch, the directories will be created, but no files will be placed in them.
/V	Verify each file transferred.

Argument Switches

/C	Convert carriage returns to NEW LINEs
/N	Do not transfer files matching this template

NOTE: @ becomes the full pathname of the working directory. Explicit directory names, such as DP0 are not changed into their corresponding AOS or AOS/VS Logical Disk Unit (LDU) names.

LIST

List the names of all files on an RDOS disk. (This command corresponds to the RDOS CLI LIST/E/A command.)

Format

XEQ RDOS LIST/DISK=rdos-diskunit
[RDOS-filename]...[template]

LIST Switches

/A	Abort on an ABORT condition.
/L	List files on LISTFILE.
/L=pathname	List files on file specified by pathname.
/T	List the contents of all directories designated in the command line. Without this switch, LIST will list the specified directories but not the files that they contain.
/V	Verify each file transferred.

Argument Switches

/N	Do not tranfer files matching this template.

NOTE: List will not list the SYS.DR of any directory unless it is explicitly listed as an RDOS filename in the command line. In addition, if there is a template in the command line, SYS.DR will not be listed regardless of whether you explicitly designate it.

LOAD

Load one or more files from an RDOS dump file. LOAD drops directory (.DR) extensions from directory files. It does not drop the .DR extension from MAP.DR files or from user files that happen to use the .DR extension.

Format

XEQ RDOS LOAD rdos-dumpfile *[rdos-filename]...*

LOAD Switches

/A	Abort on an ABORT condition.
/D	Load the new file after deleting any file with the same filename.
/L	List files on LISTFILE.
/L=pathname	List files on file specified by pathname.
/N	Do not load, just verify filenames.
/V	Verify each file transferred.

Argument Switches

/C	Convert carriage returns to NEW LINEs.
/N	Do not transfer files matching this template.

PUT

Put one or more files on an RDOS disk. You can only put files into one subdirectory or partition. PUT will access the primary partition of the RDOS disk if you omit /DIR. PUT will not create subdirectories, subpartitions, or links on the RDOS disk. Nor will it delete files on the RDOS disk.

If the AOS or AOS/VS file has a maximum index level of 0, then the resulting RDOS file will be a contiguous file. If the AOS file's maximum number of index levels is not zero, and the file will fit into one block on the RDOS disk, then the resulting RDOS file will be a sequential file. If neither of the above conditions holds, then PUT will create a random RDOS file.

Format

XEQ RDOS PUT/DISK=rdos-diskunit
[/DIR=rdos-subdirectory]
[aos-pathname]...[templates]...

PUT Switches

/A	Abort on an ABORT condition.
/DIR=rdos-subdirectory	Put files into the specified RDOS subdirectory.
/L	List files on LISTFILE.
/L=pathname	List files on file specified by pathname.
/V	Verify each file transferred.

Argument Switches

/C	Convert NEW LINES to carriage returns.
/N	Do not transfer files matching this template.

Example

) XEQ RDOS LOAD @MTB0:0 +.SR/C +.RB ⤸

Loads all the files ending in .SR and .RB from file 0 on @MTB0 into the working directory. Converts all carriage returns in the source files (.SRs) to NEW LINEs; does not convert them in the relocatable binary files (.RBs).

) XEQ RDOS LOAD/V @MTB0:1 +.SV/N ⤸

Loads all files on @MTB0:1 except files with an .SV extension.

) XEQ RDOS DUMP/V @MTB0:1 +/C ⤸

Dumps all files in the working directory to file 1 of @MTB0. Converts all of the source files' NEW LINEs to carriage returns, and lists their filenames on @OUTPUT. The dump file will be in RDOS format. All NEW LINEs will be converted, even those in .OB and .PR files.

) CREATE/DIR X ⤸
) XEQ RDOS GET/V/DISK=@DPD5 X:+.SR/C ⤸

Copies all files that have .SR extensions from RDOS subdirectory X of disk @DPD5. The directory X must exist in the working directory for this command to work.

) XEQ RDOS PUT/V/DISK=@DPD5/DIR=Y A.FR/C ⤸

Copies file A.FR from the AOS working directory to RDOS subdirectory Y on RDOS disk @DPD5 and convert NEW LINEs to CRs.

) XEQ RDOS LIST/DISK=@DPD5 Z:-.- ⤸

Lists all files matching the template -.- from RDOS subdirectory Z on RDOS disk @DPD5. Notice that the template -.- only matches files with extensions. The AOS and AOS/VS template -.- corresponds to the RDOS template -.*-.

) XEQ RDOS LIST/DISK=@DPD5 Z:+ ⤸

Lists all filenames in RDOS subdirectory Z on RDOS disk @DPD5. The AOS and AOS/VS template + corresponds to the RDOS template -.-.

) XEQ RDOS LIST/DISK=@DPD5/T ⤸

Lists the filenames of all files on RDOS disk @DPD5, including the filenames in each subdirectory.

Erroneous Examples

There are several common errors made when using the RDOS utility. Here are a few wrong examples with explanations of the errors made.

) XEQ RDOS LIST/DISK=@DPD5 U: ⤸
ERROR: FILE DOES NOT EXIST, U:

The arguments must be full RDOS pathnames. U: is not a full pathname. The correct command would have been

) XEQ RDOS LIST/DISK=@DPD5 U ⤸

If the user intended to specify the U.DR entry in the primary partition, the correct command would have been

) XEQ RDOS LIST/DISK=@DPD5 U:+ ⤸

) XEQ RDOS PUT/DISK=@DPD5/DIR=U: FOO.SR ⤸
ERROR: COULDN'T FIND SYS.DR

Here the U: is not a legal RDOS directory name. The user should have typed

) XEQ RDOS PUT/DISK=@DPD5/DIR=U FOO.SR ⤸

This will move the file FOO.SR from the working directory on the AOS disk to the subdirectory or subpartition U on the RDOS disk.

[!READ] *Pseudo-Macro*

Display text on @OUTPUT and expand to argument from @INPUT.

Format

[!READ argument *[argument]...*]

Macroname Switches

None

Argument Switches

None

Example

Given a macro including:

XEQ MASM [!READ WHICH FILE TO ASSEMBLE?]

The CLI writes the messages on the terminal and instructs the Macroassembler to assemble the filenames you type in response. The filename that you supply effectively replaces the pseudo-macro in the command line.

RELEASE *Command*

Release a logical disk from the working directory.

Format

RELEASE logical-disk *[logical-disk]...*

This command releases a previously initialized logical disk (LD). See INITIALIZE for more information about LDs. (See the *How to Generate and Run* manual for your operating system for more about LDUs.)

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) RELEASE ALPHA ↵
)
```

Releases a Logical Disk named ALPHA that you previously initialized.

RENAME *Command*

Change a file's name.

Format

RENAME pathname newname

This command renames a file you specify in pathnamee with newname. The pathname can contain the prefixes ^, =, :, and @. The newname must be a simple filename.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) FILESTATUS ↵
DIRECTORY :UDD:USER
   FILEU          CODEA
) RENAME FILEU FILEME ↵
) FILESTATUS ↵
DIRECTORY :UDD:USER
   FILEME         CODEA
)
```

REPORT *Utility*

Generate a report on the contents of the SYSLOG log file.

Format

XEQ REPORT *[pathname] ...*

This utility is described in the *How to Generate and Run AOS* (093-000217) and the *How to Generate and Run AOS/VS* (093-000243).

REVISION *Command*

Set or display a program's revision number.

Format

REVISION pathname *[field$_1$ [.field$_2$[.field$_3$[.field$_4$]]]]*

This command displays the revision number of the file named in pathname, or if you specify the pathname and field argument(s), it sets a program's revision number. By convention, Data General uses *field$_1$* for the revision number and *field$_2$* for the update number.

You may use templates in the pathname argument. *field$_1$* and *field$_2$* numbers can range from 0 to 255. You set the revision level with the .REV pseudo-op in an assembly language source program. Note that 16-bit program files (type PRG for AOS) have only two field numbers, whereas 32-bit program files (type PRV for AOS/VS) have four.

Generally, compilers and assemblers assign their own revision numbers to object files, and this revision number is carried over into the program file.

NOTE: You can only display the revision number of a UNIX type file. You can't set the revision of a UNIX file because UNIX type files are not restricted to program files, (i.e., they may also be data files) and this command is only applicable to program files.

Command Switches

/1=	{IGNORE | WARNING | ERROR | ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE | WARNING | ERROR | ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/V		Display the filename with the revision number.

REVISION (continued)

Argument Switches

None

Example

```
) REVISION JOHN ↓
00.00
) REVISION/V JOHN 0.1 ↓
JOHN 00.01
) REVISION TED ↓
00.00.01.23
)
```

First, display the revision number of a program file named JOHN in the working directory, then change the revision and display the new one.

REWIND *Command*

Rewind one or more tapes.

Format

REWIND { tapeunit / linkname } [*tapeunit* / *linkname*] ...

Specify either the same linkname you used to MOUNT the volume or the devicename on which it is mounted (tapeunit).

You may use templates in the tapeunit and linkname arguments.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) REWIND @MTB0 ↓
```

Rewind the magnetic tape on unit @MTB0.

```
) MOUNT TAPE10 MOUNT_IT_AGAIN_SAM ↓
.
.
.
) REWIND TAPE10 ↓
)
```

First requests the operator to mount a tape and create a link named TAPE10 to that tape. After performing the required operations, rewinds TAPE10.

RIC *Language*

Compile an RPG II source file using the RPG II Interpretive Compiler (DG/RIC).

Format

RIC source-pathname

The RIC macro invokes the RPG II Interpretive Compiler (DG/RIC). DG/RIC contains a debugger, analyzer, formatted dump, and high-speed compiler that produces interpretive code. Use the RPG II Interactive Editor to enter and correct the source program. Then use DG/RIC for a fast compile. Finally, when the program is debugged, compile the program using DG/ROC.

For a complete description of the RPG II programming language, see the *RPG II Reference Manual* (093-000117). For additional discussion of DG/RIC, see the *Data General/RPG II Optimizing Compiler (DG/ROC) User's Manual (AOS and AOS/VS)* (093-000279).

RIC Switches

/C	Append the compiler-generated object code to the list file (if you specified one). The object code listing includes line numbers for the lines of specifically generated code.
/CHECK	Compile the source program to a temporary file and delete the .IC, .DL, and .PR files. Use /CHECK with the /D/SOURCE/L=pathname switches for an error check and listing.
/D	Include debugger and analyzer information in the object file. This produces the same effect as a D in column 15 of the Control Card specification. The only difference is that it does not enable DEBUG calculation operations.
/D/L	Together these produce a debugger and analyzer file, a source listing, and an indicator cross reference.
/E	Suppress both page ejection and new headers when the system encounters a comment specification having asterisks in columns 7 and 8. This switch lets you run RPG II programs that use comments having lines of asterisks as separators.
/I	Inhibit conditioning indicator code optimization. This is useful for debugging, since repetitive patterns of conditioning indicators are replaced by a single test and branch.
/L	Write the source file to the current list file instead of to @OUTPUT.
/L=pathname	Write errors and source listing (if one is requested) to the file specified by pathname, instead of to @OUTPUT.
/N	Suppress the printing of notes on the compiler listing. Does not suppress warning, error, or fatal error messages.
/O=pathname	Change the names of the output files to pathname.IC, pathname.PR, and pathname.DL.
/SOURCE	Write the source program to the default @OUTPUT or to the file specified in /L=pathname.

Argument Switches

None

Example

) RIC MYPROG ↵

Compiles MYPROG.RG, creating the object file MYPROG.IC. The system does not produce a source listing, but it sends error messages to @OUTPUT.

) RIC/D/O=NEWNAME MYPROG ↵

Compiles MYPROG.RG with debugger and analyzer information in the object file. The compiler names the created files NEWNAME.IC, NEWNAME.PR, and NEWNAME.DL.

) RIC/SOURCE/L=PROG.LS MYPROG ↵

Compiles MYPROG.RG, and send a source listing and error messages to the output file PROG.LS.

ROC *Language*

Compile an RPG II source file using the RPG II Optimizing Compiler (DG/ROC).

Format

ROC source-pathname

ROC is a macro that you use to compile an RPG II source file, using the RPG II Optimizing Compiler (DG/ROC). DG/ROC produces machine instructions rather than interpretive code. As a result, DG/ROC programs run considerably faster than the same programs compiled with DG/RIC.

The ROC macro causes the following steps to be performed:

- Process the program through the DG/RIC compiler.
- Perform edit checks on the data to produce interpretive code.
- Compile the program using the DG/ROC compiler.
- Determine whether the program can be optimized.
- Produce compiled code.
- Process the file through the Link utility to produce the executable code.

The ROC macro first searches for source-pathname.RG. If that is not found, it searches for source-pathname.

For a complete description of the RPG II programming language, see the *RPG II Reference Manual* (093-000117). For more information on the ROC command line, see the *Data General/RPG II Optimizing Compiler (DG/ROC) User's Manual* (093-000279).

ROC Switches

/BUFFERS=0	Suppress buffering.
/BUFFERS=integer	Multiply the number of buffers in the file description specifications by integer (SAM files only).
/BUFFERS=−1	Allocate the maximum buffers that will fit. This is the default.
/CHECK	Execute a fast optimization check without generating optimized code.
/DSPLYRTN	Remove the suppression of DSPLY carriage return. Use this switch if DG/ROC may change the DSPLY operation, used in calculation specifications, to end with a carriage return.
/INTVAR	Store certain variables as hardware integers.
/L=pathname	Write the compiler output messages to the file specified by pathname.
/OPT=integer	Set optimization to the specified level, where integer can be 0, 1, 2, or 3. The levels are as follows: 0 Produce DG/RIC interpretive code 1 Produce DG/ROC optimized machine code 2 Increase optimization by removing redundant code 3 Turn on all the compiler tuning switches: /INTVAR, /OPT=2, and /DSPLYRTN. If you do not use any /OPT switch, the default level is 1.
/P	Generate code for a formatted dump.
/SOURCE	Write the source program to the default @OUTPUT or to the file specified in /L=pathname.
/TMPDIR=prefix	Assign high volume compiler temporary files to the specified directory. For example, if you want the files to go to directoryy :TEMP, you specify the prefix for these files, that is, :TEMP:
/TRACE	Generate code that, at execution time, will output each statement number after it is executed. If the statement changes a variable, the variable name and contents are printed.
/TRACE= line#-line# [!line#-line#] ...	Generate code that, at execution time, will trace a specified range of lines. You may specify up to five ranges of lines. Separate each range by an exclamation point: /TRACE=25–50!75–100! 125–150
/WARNOK	Ignore warnings and optimize the program.

Argument Switches

None

Example

) ROC MYPROG ↓

Compiles MYPROG.RG and produces optimized machine code.

) ROC/SOURCE/L=PROG.LS MYPROG ↓

Compiles MYPROG.RG, and sends a source listing and error messages to the output file PROG.LS.

) ROC/P/TRACE/L=PROG.LS MYPROG ↓

Compiles MYPROG.RG, and sends a listing to PROG.LS. Also, generates code that, at execution time, will write a trace to @OUTPUT and, in case of abnormal termination, will write a formatted dump to @OUTPUT.

RPG *Language*

Compile an RPG II source file.

Format

RPG source-pathname

When you invoke the RPG macro, the system actually executes the RIC macro. RPG.CLI is a link to RIC. See the description of RIC.

For a complete discussion of the RPG II programming language, see the *RPG II Reference Manual* (093-000117) and the *Data General/RPG II Optimizing Compiler (DG/ROC) User's Manual* (093-000279).

RUNTIME *Command*

Display a process's runtime information.

Format

RUNTIME [*username:procname* / *process-ID*] ...

This command displays the following runtime information about the specified process:

ELAPSED	Real-time elapsed since this process was created
CPU	Central processor time used by this process
I/O BLOCK	Number of blocks of data read or written by this process
PAGE MSECS (AOS) PAGE SECS (AOS/VS)	Number of memory pages (2K bytes) used by this process, multiplied by CPU time used (in milliseconds for AOS, and in seconds for AOS/VS)

If you include no argument, the CLI displays its own runtime information.

Command Switches

/1= { IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2= { IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/L	Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname	Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q	Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) RUNTIME 7 ↓
ELAPSED 21:44:09, CPU 0:01:47.008
I/O BLOCKS 927, PAGE MSECS 1728224
)
```

Note that, under AOS/VS, RUNTIME displays a PAGE SECS value rather than a PAGE MSECS value.

SCOM *Utility*

Compare two ASCII text files.

Format

XEQ SCOM sourcefile$_1$ sourcefile$_2$

This program scans each line from both files. If it finds differences, it outputs either the difference or a message (see SCOM Switches). The program then attempts to get back into synchronization. Synchronization is defined as finding n lines in a row that match (where n is called the matchsize). By default, the matchsize is automatically set to four.

For example, the following sequence could be found several times in one source file:

```
ISZ     ?ORTN,3         ;INDICATE SUCCESS
RTN                     ;RETURN
```

Thus, SCOM would get back into synchronization if the next two lines following the above series matched.

The maximum number of characters per line for either file is 133, including the NEW LINE terminator, where a line is any string ending in an ASCII NEW LINE. The maximum number of lines on a page for either file is 32767, where a line is any string ending in an ASCII NEW LINE. The maximum number of pages for either file is also 32767, where a page is any string ending in an ASCII form feed.

SCOM Switches

/EOL	The end-of-line character is treated as significant. If you omit this switch, SCOM ignores EOL characters and blank lines.
/L	Write the list of differences to current @LIST file. If you omit this switch, the program doesn't list the differences. Instead, it outputs the message *FILES DIFFER STARTING AT LINE xxx/xxx* if the files differ, or a CLI prompt if the files match.
/L=filename	Write differences to filename.
/MS=number	Set matchsize to the specified value (number). If you omit this switch, the default is 4.

Argument Switches

None

Example

```
) XEQ SCOM MYFILE YOURFILE ↵
```

SCREENEDIT *Command*

Set or display the current SCREENEDIT mode.

Format

SCREENEDIT [*ON* / *OFF*]

If SCREENEDIT is ON (the default in interactive mode), you can modify what you type into your terminal by using cursor control characters. In batch jobs, the default is SCREENEDIT OFF. Note that SCREENEDIT ON is only valid for display terminals.

NOTE: With SCREENEDIT ON, the maximum line length is 76 characters, as opposed to 128 characters when SCREENEDIT is OFF. If you type a command line that exceeds 76 characters, the CLI will automatically continue the line and issue the continuation line prompt &). All subsequent characters are placed in the continuation line. If you want to change the first line, you must first abort the current line with a CTRL-C CTRL-A sequence, and then rewrite the line. Also, Insertion mode (CTRL-E) is terminated as soon as the number of characters originally input plus the newly inserted ones totals 76.

Control Characters

CTRL-A	Move to the end of the character string
CTRL-B	Move to the end of the previous word
CTRL-E	Enter/exit the insert character mode
CTRL-F	Move to the beginning of the next word
CTRL-H	Move to the beginning of the character string (The HOME key has the same effect)
CTRL-I	Insert a tab (The TAB key has the same effect)
CTRL-K	Erase everything to the right of the cursor (The ERASE EOL key has the same effect)
CTRL-X	Move to the right one character (The → key on the function keypad has the same effect)
CTRL-Y	Move to the left one character (The ← key on the function keypad has the same effect)

Command Switches

/1=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by **pathname** instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/P		Set the current SCREENEDIT mode to the previous environment's SCREENEDIT mode (no arguments are allowed with this switch).

Argument Switches

None

Example

```
) SCREENEDIT ON )
) ANT CHARACTER STRING. (CTRL-H) (CTRL-X)
    (CTRL-X) Y
```

Typing CTRL-H returns the cursor to the beginning of the string. Typing CTRL-X twice positions the cursor at the T in the first word. After you've corrected the string by replacing the T with Y, it reads:

```
) ANY CHARACTER STRING.
```

SEARCHLIST *Command*

Set or display the search list setting.

Format

SEARCHLIST *[pathname]...*

If you use the /P command switch, the CLI displays the previous environment's search list; otherwise, the CLI displays the current search list. See Chapters 2 and 4 for an explanation of search lists.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/K		Delete the current search list, if any exists (no arguments allowed).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname @OUTPUT.
/P		Set search list to previous environment's search list (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) SEARCHLIST ↵
:PER,:UTIL,:
) PUSH ↵
) SEARCHLIST :UDD:HENRY,:PER,:UTIL,: ↵
) SEARCHLIST ↵
:UDD:HENRY,:PER,:UTIL,:
) SEARCHLIST/P ↵
) SEARCHLIST ↵
:PER,:UTIL,:
)
```

Displays the current search list and then pushes a level. Changes the current search list and displays it. Then sets the search list to its original value and displays it again.

[!SEARCHLIST] *Pseudo-Macro*

Expand to the search list.

Format

[!SEARCHLIST]

This pseudo-macro does not accept arguments.

Macroname Switches

/P Expands to the previous environment's search list.

Argument Switches

None

Example

) SEARCHLIST :UDD:MDIR,[!SEARCHLIST] ↓
)

Evaluates the !SEARCHLIST pseudo-macro, then sets the search list to the resulting argument string. The effect is the addition of :UDD:MDIR to the current search list.

SED *Utility*

Edit an ASCII text file.

Format

XEQ SED *[pathname]*

This command calls the SED text editor program. If the file you specify in *pathname* does not exist, SED asks:

DO YOU WANT pathname TO BE CREATED?

Answer Y to create the file. SED then displays its prompt (*).

If you include *pathname*, and the file exists, SED opens it for editing and displays the * prompt.

See the *SED Text Editor User's Manual* (093-000249) for more information.

SED Switches

/ED=directory-name	Store the .ED file, which contains formatting settings, in the directory specified by directory-name.
/NO_ED	Do not create an .ED file.
/NO_SCREEN	Do not automatically update the screen. This switch is useful for a console with a low baud rate or for a hard-copy terminal.
/PROFILE=pathname	Begin the editing session by executing the SED commands contained in pathname.
/WORK=directory-name	Store all temporary SED files in the directory specified by directory-name.

Argument Switches

None

Example

) XEQ SED/ED=NEAT/WORK=:FIX_HEAD REPORT ↓

Invokes SED to edit the file REPORT. Places the file containing formatting setting in the directory NEAT, and place all temporary files on the fixed-head disk.

SEND *Command*

Send a message to a terminal.

Format

SEND { process-ID / username:procname / consolename } message

Use this command to send a message to a process's terminal. The target process can be any process with a terminal. A procname can be either a simple or complete process. A complete procname is in the form

username:process

for example, BOOTHBY:CON7. A consolename may begin with either :PER: or the @ prefix. Do not try to include commas, tabs, semicolons, or control characters in your messages. Commas and tabs become spaces when the system sends your message. Semicolons terminate your message, and the system cannot send control characters.

If you send a message and the target process doesn't receive it, that process may have disabled message reception. You can disable message reception by using the /NRM switch on the CHARACTERISTICS command. You can use templates for the destination argument.

Command Switches

/1= { IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2= { IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/I	Send each of the following input lines as a separate message until you reach a line containing a single).
/L	Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname	Write CLI output to the file specified by pathname instead of to @OUTPUT.
/M	Send each line of the current macro file as a separate message. You must end the macro sequence with a single).
/Q	Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) SEND 2 PLEASE BRING UP LPT1 ↵
)
```

Sends the message to the system operator.

```
) SEND @CON- SYSTEM WILL SHUT DOWN & ↵
&)AT MIDNIGHT. ↵
)
```

Sends the message to all consoles that are running a process.

FROM PID 8: IS IT OK FOR ME TO PRINT A LONG MANUSCRIPT?

```
) SEND 8 GO AHEAD. ↵
)
```

Replies to a message received from process 8.

[!SIZE] *Pseudo-Macro*

Expand to a file's length in bytes.

Format

[!SIZE pathname]

This pseudo-macro expands to the byte length of the file you specify (by its pathname). If the pathname does not exist, the pseudo-macro returns 0.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE THE LENGTH OF FOO IS [!SIZE FOO] ↓
THE LENGTH OF FOO IS 14
)
```

SLB *Utility*

Build a shared library (AOS only).

Format

XEQ SLB/O=shared-routine-name library-number programname *[programname]...*

Call the Shared Library Builder utility to build a shared library routine from one or more executable programs. The SLB searches for programnames with the .PR extension, but you need not type the extension. The SLB always appends the extension .SL to shared-routine-name. Library-number is a number in the range 2 through 63 that you want the new routine to have (numbers 0 and 1 are system reserved). See the *AOS Shared Library Builder User's Manual* (093-000191) for more on the SLB.

SLB Switches

/L	Send the listing to the current @LIST file.
/L=pathname	Send the listing to the file specified by pathname.

Argument Switches

/O	This is the output filename; /O is a mandatory switch.

Example

```
) XEQ SLB/L=@LPT/O=GEOM5 SINE COSINE &↓
&) TANGENT ↓
```

Creates shared library GEOM.SL from executable programs SINE.PR, COSINE.PR, and TANGENT.PR. Because GEOM.SL is a shared routine, many users can access it from their own programs.

SORT/MERGE *Language*

Invoke the Sort/Merge utility

Format

```
{SORT  }
{      }  [INTO outfile-pathname
{MERGE }     [FROM infile-pathname] ...]
```

SORT and MERGE are macros that invoke the Sort/Merge utility. This general-purpose utility manipulates record order and content. It gives you the power to sort and copy records; to merge multiple files into a single file; to edit records in files; to delete duplicate records during a sort or merge operation; and to delete records according to any criteria you specify.

To take full advantage of the utility's features, you can use a *command file* . The command file contains statements telling the utility where to find the records to be sorted or merged, where to send the sorted or merged records, and how to perform the sort or merge. Since you can save a command file on disk, a single command file can be used for many repeated tasks.

You must declare the input and output files. You can declare the files in the command file, or you use the INTO FROM phrase to declare them in the command line. You can also declare the output file in the command line while declaring some or all of the input files in the command file.

For a complete discussion of the Sort/Merge utility, see the *Sort/Merge with Report Writer User's Manual* (093-000155).

Sort/Merge Switches

/C	Indicate that you intend to enter a command file from your terminal.
/C=pathname	Use the file specified by pathname as the command file.
/L	Write statistical output and any error messages to the current list file.
/L=pathname	Write statistical output and any error messages to the file specified by pathname.
/N	Suspend execution of the imperative. The utility still checks the syntax of the command file statements.
/O	Delete the output file, if it exists, and recreate it with the results of the Sort/Merge process.
/S	Suppress statistical output.
/T=pathname	Save the command file that will be typed in at the terminal. The command file is saved in the file specified by pathname.

Example

```
) SORT/C=REORDER INTO TEACHERS FROM& )
&) REGISTER_6 )
```

Invokes Sort/Merge to execute the command file REORDER. Sort/Merge sorts a copy of the records in input file REGISTER_6, and sends the sorted records to output file TEACHERS.

SPACE *Command*

Set or display the amount of disk space in a control-point directory or logical disk.

Format

SPACE [*control-point-directory [new-max-size]* | *logical-disk*]

SPACE returns the maximum, current, and remaining disk space in 512-byte disk blocks. See the appropriate programmer's manual for more information about control point directories and logical disks.

You may use templates for the directory argument.

Command Switches

/1=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS1 to the specified severity level for this command.
/2=	(IGNORE | WARNING | ERROR | ABORT)	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/V		Display control-point-directory or logical-disk name.

Argument Switches

None

Example

```
) SPACE/V ↓
= MAX 370889, CUR 304097, REM 66792
)
```

Displays disk space in =, the working directory.

```
) SPACE : ↓
MAX 37000, CUR 18000, REM 19000
)
```

Displays disk space in :, the root directory.

SPEED *Utility*

Edit an ASCII text file.

Format

XEQ SPEED *[pathname]*

SPEED is a text editor. You can use it to create a new source file or edit an existing source file. If the file you specify in *pathname* does not exist, SPEED asks:

Create new file?

Answer Y to create the file. SPEED then displays its prompt (*!*).

For more information, consult the *SPEED Text Editor (AOS and AOS/VS) User's Manual* (093-000197).

SPEED Switches

/I=pathname	Take the SPEED commands from the file specified in pathname, rather than from @INPUT, (usually the keyboard).
/D	Display text automatically. If you include both /D and /I, /D is ignored.

Argument Switches

None

Example

) XEQ SPEED/I=PROCESS_REP.SCF REPORT1 ↓

Invokes SPEED to edit the file REPORT1. The editing is not an interactive session, but is performed by executing the SPEED commands contained in the file PROCESS_REP.SCF.

SQUEEZE *Command*

Set or display the SQUEEZE setting.

Format

SQUEEZE [*ON* / *OFF*]

When SQUEEZE mode is ON, the CLI outputs each sequence of two or more tabs or spaces as a single space. (The system never squeezes output from the TYPE command.) You can turn SQUEEZE mode ON for the processing of any single CLI command by appending the /Q switch to the command name. See SQUEEZE in Chapter 3.

Command Switches

/1=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set the current SQUEEZE mode to the previous environment's SQUEEZE mode (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

SQUEEZE (continued)

Argument Switches

None

Example

```
) SQUEEZE ↓
OFF
) SQUEEZE ON ↓
) SQUEEZE ↓
ON
)
```

Displays the current SQUEEZE setting, then sets SQUEEZE to ON, and finally displays the new SQUEEZE setting.

STRING *Command*

Set or display STRING setting.

Format

STRING *[argument]...*

This command without an argument displays the contents of the CLI STRING buffer. With an argument, it sets STRING's values to the argument. The STRING buffer can hold up to 127 characters. We describe STRING in Chapter 4.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/K		Set STRING to null (no arguments allowed).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set STRING to previous environment's STRING (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) STRING ↓
THIS,IS,A,STRING
) STRING  NEW ONE ↓
) STRING ↓
NEW,ONE
) STRING/K ↓
) STRING ↓
)
```

First, displays STRING, then sets it to a new string. Kills the current STRING, and displays STRING again (a null line).

[!STRING] *Pseudo-Macro*

Expand to the STRING setting.

Format

[!STRING]

This pseudo-macro does not accept arguments.

Macroname Switches

/P Returns the previous environment's STRING.

Argument Switches

None

Example

```
) WRITE THE CURRENT STRING IS [!STRING] ↓
THE CURRENT STRING IS CURRENT_STRING
) WRITE THE PREVIOUS STRING IS [!STRING/P] ↓
THE PREVIOUS STRING IS PREVIOUS_STRING
)
```

SUPERPROCESS *Command*

Set or display the SUPERPROCESS setting.

Format

SUPERPROCESS [*ON* / *OFF*]

Only privileged users can set Superprocess to ON. See Chapter 4 for more information about Superprocess.

The CLI precedes each prompt with a plus sign (+) when you have SUPERPROCESS turned ON. If you enable both SUPERPROCESS and SUPERUSER (see following description), the CLI displays a number sign (#) before the prompt. Given the initial default prefix (a right parenthesis), the prompts look like this:

Prompt	Superprocess	Superuser
)	OFF	OFF
*)	OFF	ON
+)	ON	OFF
#)	ON	ON

Command Switches

/1= {IGNORE / WARNING / ERROR / ABORT}
Set CLASS1 to the specified severity level for this command.

/2= {IGNORE / WARNING / ERROR / ABORT}
Set CLASS2 to the specified severity level for this command.

/L
Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname
Write CLI output to the file specified by pathname instead of to @OUTPUT.

/P
Set the current Superprocess mode to previous environment's Superprocess mode (no arguments allowed).

/Q
Set SQUEEZE to ON for this command.

Arguments Switches

None

Example

```
) SUPERPROCESS ↓
OFF
) SUPERPROCESS ON ↓
+)SUPERPROCESS ↓
ON
+) ↓
```

First, displays the current Superprocess setting. Second, turn Superprocess ON. (Note that with Superprocess turned ON, the CLI outputs the prompt +) in the left margin). Last, displays the current Superprocess setting.

SUPERUSER *Command*

Set or display the SUPERUSER setting.

Format

SUPERUSER [*ON* / *OFF*]

Normally, you have access (ACL) rights only to files created in and under your user directory, and read access to files in :UTIL. If your user profile gives you Superuser privilege, you can set Superuser to ON, enabling you to have access to every file in the system, without regard to access control lists (ACLs). See Chapter 4 for more information about Superuser.

The CLI precedes each prompt with an asterisk (*) when you have Superuser turned ON. If you enable both Superuser and Superprocess, the CLI displays a number sign (#) before the prompt. Given the initial default prefix (a right parenthesis), the prompts look like this:

Prompt	Superprocess	Superuser
)	OFF	OFF
*)	OFF	ON
+)	ON	OFF
#)	ON	ON

Command Switches

/1= {IGNORE / WARNING / ERROR / ABORT}
Set CLASS1 to the specified severity level for this command.

/2= {IGNORE / WARNING / ERROR / ABORT}
Set CLASS2 to the specified severity level for this command.

/L
Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname
Write CLI output to the file specified by pathname instead of to @OUTPUT.

/P
Set the current Superuser mode to previous environment's Superuser mode (no arguments allowed).

/Q
Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) SUPERUSER ↵
OFF
) SUPERUSER ON ↵
*)SUPERUSER ↵
ON
*)
```

First, displays the current Superuser setting. Second, turns Superuser ON. (Note that with Superuser turned ON, the CLI outputs the prompt *) in the left margin). Finally, displays the current SUPERUSER setting.

SWAT *Utility*

Invoke the SWAT interactive debugger.

Format

XEQ SWAT program-pathname
[program-argument]...

The SWAT debugger is a high-level, interactive symbolic debugging system for high-level language programs. Using SWAT, you can check a program's correctness at the level of the source language, rather than at the assembly or machine-language level.

To use AOS SWAT under AOS/VS, invoke the program called SWAT16.PR. SWAT16 accepts the same switches as SWAT.

For a complete description of SWAT, see the *SWAT® Debugger User's Manual* (093-000258).

SWAT Switches

/AUDIT*[=pathname]*	Maintain an audit of this SWAT session. If you specify a pathname, SWAT writes the audit information to that file. Otherwise, SWAT will write the information to program-pathname.AU.
/CONSOLE=consolename	Assign a new terminal for the program being debugged.
/DATA=pathname	Associate a new filename with @DATA.
/DEBUG	(AOS only) Begin execution in the Symbolic Debugger, from which you can move into SWAT.
/INPUT=pathname	Associate a new filename with @INPUT.
/LIST=pathname	Associate a new filename with @LIST.
/OUTPUT=pathname	Associate a new filename with @OUTPUT.

Argument Switches

Use any argument switches appropriate for the program specified in program-pathname.

Example

```
) XEQ SWAT/DATA=DEBUG.DATA& )
&)/LIST=DEBUG.LIST MYPROG )
```

Invokes SWAT to debug program MYPROG. For debugging, uses DEBUG.DATA where the program calls for the generic @DATA file, and DEBUG.LIST where it calls for the generic @LIST file.

SYSID *Command*

Set or display the unique system identifier.

Format

SYSID *[argument]* ...

This command displays or sets the system identifier of your computer system. Only PID 2 can set the system identifier.

Command Switches

/1=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) SYSID )
R E M U L A C
```

This displays the current system identifier.

SYSINFO *Command*

Display system information (AOS only).

Format

SYSINFO

This command displays the following information about the current system environment:

- system revision
- system memory
- master logical disk
- system identifier
- current system

Command Switches

/1=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE / WARNING / ERROR / ABORT}	Set CLASS2 to the specified severity level for this command.
/L=		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

SYSINFO (continued)

Argument Switches

None

Example

```
) SYSINFO ⟩
SYSTEM REVISION: 03.30
SYSTEM MEMORY: 1023 PAGES
MASTER LOGICAL DISK: ROOT.7.25.79
SYSID: R E M U L A C
CURRENT SYSTEM: :SYSGEN:REMULAC.PR
)
```

This displays current system information.

```
) SYSINFO ⟩
SYSTEM REVISION: 03.30
SYSTEM MEMORY: 1023 PAGES
MASTER LOGICAL DISK: ROOT.7.25.79
SYSID: R E M U L A C
"INSTALLED SYSTEM"
```

If the current system is the installed system, this is displayed.

SYSLOG *Command*

Set or display the SYSLOG setting.

Format

SYSLOG *[filename]*

Normally, the system writes the following information to the log file:

- Information about system users, such as when they log on, when they log off, what devices they use, CPU usage, and the size of main memory allocated to them.
- Information about peripheral devices, such as type and number of errors they encounter.

This is a privileged command; only the initial CLI process (PID 2) may issue the SYSLOG call.

Command Switches

/1=	IGNORE | WARNING | ERROR | ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE | WARNING | ERROR | ABORT	Set CLASS2 to the specified severity level for this command.
/ERROFF		(AOS only) Inhibit sending soft device error messages to the operator's console.
/ERRON		(AOS only) Enable sending of soft device error messages to the operator's console.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.
/START		Start system log file.
/STOP		Stop system log file.

Argument Switches

None

Example

```
) SYSLOG/START ↓
)
```

Starts recording in the system log file.

```
) SYSLOG ↓
ON
)
```

SYSLOG with no arguments returns the current state of the log file.

[!SYSTEM] *Pseudo-Macro*

Expand to the name of the operating system.

Format

```
[!SYSTEM]
```

This pseudo-macro does not accept arguments.

Macroname Switches

None

Argument Switches

None

Example

Given a macro including:

```
[!EQUAL,[!SYSTEM],AOS]
      .
      .
      .
[!END]
[!EQUAL,[!SYSTEM],AOS/VS]
      .
      .
      .
[!END]
```

The commands between the [!EQUAL,[!SYSTEM],AOS] and the first !END will be executed only when the system is running AOS. The commands between the [!EQUAL,[!SYSTEM],AOS/VS] and the second !END will be executed only when the system is running AOS/VS.

TERMINATE *Command*

Terminate an inferior process.

Format

TERMINATE { username:procname / process-ID } [*username:procname / process-ID*] ...

You must supply the **process-ID** or the **procname** of an inferior process unless Superprocess is ON. **procname** can be either a simple or complete process name.

Command Switches

/1= { IGNORE / WARNING / ERROR / ABORT } Set CLASS1 to the specified severity level for this command.

/2= { IGNORE / WARNING / ERROR / ABORT } Set CLASS2 to the specified severity level for this command.

/BREAKFILE *[=pathname]* Produce a breakfile in the working directory at the time of termination. When you use /BREAKFILE to call TERMINATE, the system creates a break file with a default name (a filename with a .BRK filename extension). When you call TERMINATE with /BREAKFILE= pathname, the break file will have the specified name.

/L Write CLI output to the current list file instead of to @OUTPUT.

/L=pathname Write CLI output to the file specified by **pathname** instead of to @OUTPUT.

/Q Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) PROCESS SMITH:PROGA ↵
PID 17
.
.
.
) TERMINATE 17 ↵
)
```

First, creates a swappable son process that runs concurrently with the CLI. (The CLI displays the PID of the new process.) Then terminates the process.

TIME *Command*

Set or display the current system time.

Format

TIME *[new-time]*

This command displays or sets the system time. Only the master CLI (PID2) can set the time. Using a 24-hour clock, *new-time* is in the following format:

hours *[minutes seconds]*

minutes and *seconds* are optional. You can use spaces or colons to separate entries.

Command Switches

/1=	IGNORE, WARNING, ERROR, ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE, WARNING, ERROR, ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) TIME ↵
19:30:45
) TIME 8 45 ↵
)
```

Displays the system time, then sets the time to 8:45 a.m.

[!TIME] *Pseudo-Macro*

Expand to the current system time.

Format

[!TIME]

This pseudo-macro does not accept arguments.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE IT IS NOW [!TIME]. ↵
IT IS NOW 03:47:21.
)
```

TRACE *Command*

Set or display the current trace mode.

Format

TRACE

Each tracing mode can be turned on and off independently through the use of switches. With tracing on, you can see the actual command line as the CLI processes it. The symbol for command tracing is ***, for macro tracing ###, and for pseudo-macro tracing +++.

If you use the /LOG switch and a log file is open, then the trace output will go to the log file. Otherwise, the trace output will go to @OUTPUT.

Command Switches

/1=	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS1 to the specified severity level for this command.
/2	{IGNORE, WARNING, ERROR, ABORT}	Set CLASS2 to the specified severity level for this command.
/COMMAND		Specify command tracing.
/KILL		Turn all tracing modes off (no arguments (allowed).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/LOG		Send trace output to the log file.
/MACRO		Specify macro tracing.
/OFF		Turn off the tracing modes specified by the TRACE switches that follow.
/ON		Turn on the tracing modes specified by the TRACE switches that follow.
/PREVIOUS		Set trace mode to previous trace mode (no arguments allowed).
/PSEUDO		Specify pseudo-macro tracing.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) TRACE/MACRO ↓
) TRACE/COMMAND ↓
) TRACE ↓
***TRACE
/COMMAND/MACRO
)
```

First turns on macro tracing, and then turn on command tracing. Then displays the trace modes. Note that the three asterisks before TRACE are the command tracing symbols.

TREE *Command*

Display a process's family tree.

Format

TREE [*username:procname* | *process-ID*] ...

This command displays a process's father and sons (if any exist). If you omit the argument, TREE displays the CLI's tree.

Command Switches

/1=	{ IGNORE | WARNING | ERROR | ABORT }	Set CLASS1 to the specified severity level for this command.
/2=	{ IGNORE | WARNING | ERROR | ABORT }	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) TREE 7 ↓
PID: 7, FATHER: 4, SONS: 8   12   13
) TREE OP:EXEC ↓
PID: 4, FATHER: 2, SONS: 7   9   10   11
)
```

Displays PID 7's tree. Then, displays EXEC's tree.

TYPE *Command*

Type the contents of a file.

Format

TYPE pathname *[pathname]...*

This command displays the contents of the file(s) named in pathname on your terminal screen. This enables you to read text files. You can only read files to which you have read access, unless you have the Superuser privilege. You may use templates for the pathname argument(s).

SQUEEZE mode doesn't compress output from TYPE.

While the CLI will attempt to type large, fixed length files (i.e. 25,000 bytes), there may not be enough stack space for the CLI to type files greater than 20,000 bytes.

Command Switches

/1=	{ IGNORE | WARNING | ERROR | ABORT }	Set CLASS1 to the specified severity level for this command.
/2=	{ IGNORE | WARNING | ERROR | ABORT }	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command (This switch has no effect on the TYPE command).
/V		Display the title and record type of the file before typing it. If the record type is fixed, the CLI also displays the record length.

TYPE (continued)

Argument Switches

None

Example

```
) TYPE MYFILE )
     . (contents of MYFILE)
     .
     .
) TYPE/2=ERROR FILE1 FILE2 FILE3 )
     . (contents of FILE1, FILE2, FILE3)
     .
     .
)
```

Displays MYFILE on the terminal. Then, sets CLASS2 exceptional conditions to ERROR and types FILE1, FILE2, and FILE3. If any of the named files do not exist, the CLI will display an ERROR message and will cease typing. It will not type files whose names appear to the right of the nonexistent filename.

[!UADD] *Pseudo-Macro*

Expand to the sum of two numbers.

Format

[!UADD argument$_1$ argument$_2$]

The pseudo-macro requires two arguments, which must evaluate to unsigned decimal integers. Both arguments may be double precision. Double precision is in the range 0 to 4,294,967,295.

The value returned is the sum of argument$_1$ plus argument$_2$. The sum must be within the double-precision range. If the two arguments produce a sum greater than this range, the value returned will be equivalent to the actual sum modulo 4,294,967,296.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!UADD 5 6] )
11
)
```

Adds two integers and display the sum.

Given a macro containing:

```
[!ULE,[!UADD,[!SIZE FILE1],[!SIZE FILE2],10000]
```

Evaluates the two !SIZE pseudo-macros, and then the !UADD pseudo-macro, which returns the sum of the sizes of the two files. If the size does not exceed the indicated number of bytes, execute the code between the !ULE pseudo-macro and the next !ELSE or !END.

[!UDIVIDE] *Pseudo-Macro*

Expand to the quotient of two numbers.

Format

[!UDIVIDE argument$_1$ argument$_2$]

The pseudo-macro requires two arguments, which must evaluate to unsigned decimal integers. argument$_1$ may be double precision, which is in the range 0 to 4,294,967,295. argument$_2$ must be single precision, which is in the range 1 to 65,535.

The value returned is the integer result of argument$_1$ divided by argument$_2$. Note that argument$_2$ cannot equal 0.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!UDIVIDE 10 3] )
3
)
```

Divides the first integer by the second, and displays the quotient.

The macro TIP.CLI, which contains the following lines, prompts for input and computes 15% of the input amount:

```
PUSH
VAR0 [!READ TOTAL CHECK IN CENTS?,,,,]
WRITE TIP IS [!UDIVIDE,[!UMULTIPLY,15,[!VAR0]],100]
POP
```

This macro pushes a level, prompts for input, and assigns the value to a variable. It then multiplies the amount by .15: that is, multiplies the amount by 15, and then divides the product by 100. Finally, the macro outputs this final quotient.

[!UEQ] *Pseudo-Macro*

Include input conditionally.

Format

[!UEQ argument$_1$ argument$_2$]

!UEQ compares the two arguments. If they are equal, the CLI executes the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI doesn't execute the input between the !ELSE and the !END.

This pseudo-macro begins a sequence of text that the CLI is to execute conditionally. You must end the sequence with the !END pseudo-macro. The sequence can also include the !ELSE pseudo-macro.

The !UEQ pseudo-macro must always have two arguments, which must evaluate to unsigned decimal integers. Both arguments must be in the range 0 to 4,294,967,295.

If the arguments are not equal, the CLI does not execute the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI executes the input between the !ELSE and the !END.

See Chapter 5 for a discussion of conditional pseudo-macros.

Macroname Switches

None

Argument Switches

None

Example

Given a macro containing:

```
[!UEQ,[!VAR0],17]
WRITE VAR0 = 17
[!ELSE]
WRITE VAR0 = [!VAR0]
[!END]
```

This macro writes *VAR0 = 17* if the current value of VAR0 is 17. Otherwise, it writes *VAR0 = n*, where n is the current value of VAR0.

You can also code the macro as follows:

```
WRITE VAR0 &
=[!UEQ,[!VAR0],17]17[!ELSE][!VAR0][!END]
```

[!UGE] *Pseudo-Macro*

Include input conditionally.

Format

[!UGE argument$_1$ argument$_2$]

If argument$_1$ is greater than or equal to argument$_2$, the CLI executes the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI doesn't execute the input between the !ELSE and the !END.

This pseudo-macro begins a sequence of text that the CLI is to execute conditionally. You must end the sequence with the !END pseudo-macro. The sequence can also include the !ELSE pseudo-macro.

The !UGE pseudo-macro must always have two arguments, which must evaluate to unsigned decimal integers. Both arguments must be in the range 0 to 4,294,967,295.

If argument$_1$ is less than argument$_2$, the CLI does not execute the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI executes the input between the !ELSE and the !END.

See Chapter 5 for a discussion of conditional pseudo-macros.

Macroname Switches

None

Argument Switches

None

Example

Given a macro containing:

```
[!UGE,[!VAR7],10]
WRITE VAR7 IS GREATER THAN OR EQUAL TO 10
[!ELSE]
WRITE VAR7 IS LESS THAN 10
[!END]
```

This macro writes the first message if the current value of VAR7 is greater than or equal to 10. Otherwise, it writes the second message.

You can also code the macro as follows:

```
[!UGE,[!VAR7],10]WRITE VAR7 IS GREATER THAN &
OR EQUAL TO 10
OR[!ELSE]WRITE VAR7 IS LESS THAN 10[!END]
```

Note that there are no spaces between the bracketed !UGE statement and its WRITE command, nor between the !ELSE statement and its WRITE command.

[!UGT] *Pseudo-Macro*

Include input conditionally.

Format

[!UGT argument$_1$ argument$_2$]

If argument$_1$ is greater than argument$_2$, the CLI executes the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI doesn't execute the input between the !ELSE and the !END.

This pseudo-macro begins a sequence of text that the CLI is to execute conditionally. You must end the sequence with the !END pseudo-macro. The sequence can also include the !ELSE pseudo-macro.

The !UGT pseudo-macro must always have two arguments, which must evaluate to unsigned decimal integers. Both arguments must be in the range 0 to 4,294,967,295.

If argument$_1$ is less than or equal to argument$_2$, the CLI does not execute the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI executes the input between the !ELSE and the !END.

See Chapter 5 for a discussion of conditional pseudo-macros.

Macroname Switches

None

Argument Switches

None

Example

Given a macro containing:

```
[!UGT,[!VAR1],11]
WRITE VAR1 IS GREATER THAN 11
[!ELSE]
WRITE VAR1 IS LESS THAN OR EQUAL TO 11
[!END]
```

This macro writes the first message if the current value of VAR1 is greater than 11. Otherwise, it writes the second message.

You can also code the macro as follows:

```
[!UGT,[!VAR1],11]WRITE VAR1 IS GREATER THAN 11
[!ELSE]WRITE VAR1 IS LESS THAN OR EQUAL &
TO 11[!END]
```

Note that there are no spaces between the bracketed !UGT statement and its WRITE command, nor between the !ELSE statement and its WRITE command.

[!ULE] *Pseudo-Macro*

Include input conditionally.

Format

[!ULE argument$_1$ argument$_2$]

If argument$_1$ is less than or equal to argument$_2$, the CLI executes the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI doesn't execute the input between the !ELSE and the !END.

This pseudo-macro begins a sequence of text that the CLI is to execute conditionally. You must end the sequence with the !END pseudo-macro. The sequence can also include the !ELSE pseudo-macro.

The !ULE pseudo-macro must always have two arguments, which must evaluate to unsigned decimal integers. Both arguments must be in the range 0 to 4,294,967,295.

If argument$_1$ is greater than argument$_2$, the CLI does not execute the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI executes the input between the !ELSE and the !END.

See Chapter 5 for a discussion of conditional pseudo-macros.

Macroname Switches

None

Argument Switches

None

Example

Given a macro containing:

```
[!ULE,[!VAR5],2]
WRITE VAR5 IS LESS THAN OR EQUAL TO 2
[!ELSE]
WRITE VAR5 IS GREATER THAN 2
[!END]
```

This macro writes the first message if the current value of VAR5 is less than or equal to 2. Otherwise, it writes the second message.

You can also code the macro as follows:

```
[!ULE,[!VAR5],2]WRITE VAR5 IS LESS THAN OR &
EQUAL TO 2
[!ELSE]WRITE VAR5 IS GREATER THAN 2[!END]
```

Note that there are no spaces between the bracketed !ULE statement and its WRITE command, nor between the !ELSE statement and its WRITE command.

[!ULT] *Pseudo-Macro*

Include input conditionally.

Format

[!ULT argument$_1$ argument$_2$]

If argument$_1$ is less than argument$_2$ the CLI executes the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI doesn't execute the input between the !ELSE and the !END.

This pseudo-macro begins a sequence of text that the CLI is to execute conditionally. You must end the sequence with the !END pseudo-macro. The sequence can also include the !ELSE pseudo-macro.

The !ULT pseudo-macro must always have two arguments, which must evaluate to unsigned decimal integers. Both arguments must be in the range 0 to 4,294,967,295.

If argument$_1$ is greater than or equal to argument$_2$, the CLI does not execute the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI executes the input between the !ELSE and the !END.

See Chapter 5 for a discussion of conditional pseudo-macros.

Macroname Switches

None

Argument Switches

None

Example

Given a macro containing:

```
[!ULT,[!VAR2],13]
WRITE VAR2 IS LESS THAN 13
[!ELSE]
WRITE VAR2 IS GREATER THAN OR EQUAL TO 13
[!END]
```

This macro will write the first message if the current value of VAR2 is less than 13. Otherwise, it will write the second message.

You can also code the macro as follows:

```
[!ULT,[!VAR2],13]WRITE VAR2 IS LESS THAN 13
[!ELSE]WRITE VAR2 IS GREATER THAN OR &
EQUAL TO 13[!END]
```

Note that there are no spaces between the bracketed !ULT statement and its WRITE command, nor between the !ELSE statement and its WRITE command.

[!UMODULO] *Pseudo-Macro*

Expand to the result of the modulus operation on two numbers.

Format

[!UMODULO argument$_1$ argument$_2$]

The pseudo-macro requires two arguments, which must evaluate to unsigned decimal integers. argument$_1$ may be double precision, which is in the range 0 to 4,294,967,295. argument$_2$ must be single precision, which is in the range 1 to 65,535.

The value returned is the integer result of argument$_1$ modulo argument$_2$. For unsigned integers, this value is equivalent to the remainder produced by dividing argument$_1$ by argument$_2$. Note that argument$_2$ cannot equal 0.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!UMODULO 10 3] ↓
1
)
```

Performs the modulus operation on two integers, and displays the results.

The macro DIVIDE.CLI, which contains the following command lines, divides the first argument by the second argument, and displays the quotient and the remainder (that is, the result of the modulus operation):

```
WRITE THE QUOTIENT IS [!UDIVIDE,%1%,%2%]
WRITE THE REMAINDER IS [!UMODULO,%1%,%2%]
```

[!UMULTIPLY] *Pseudo-Macro*

Expand to the product of two numbers.

Format

[!UMULTIPLY argument$_1$ argument$_2$]

The pseudo-macro requires two arguments, which must evaluate to unsigned decimal integers. argument$_1$ may be double precision, which is in the range 0 to 4,294,967,295. argument$_2$ must be single precision, which is in the range 0 to 65,535.

The value returned is the result of argument$_1$ multiplied by argument$_2$. The product must be within the double precision range. If the two arguments create a product greater than this range, the value returned will be equivalent to the actual product modulo 4,294,967,296.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!UMULTIPLY 66777 92] ↓
6143484
)
```

Multiplies two integers and displays the product.

The macro CENT.CLI, which contains the following command lines, takes as its argument an integer value in degrees Fahrenheit (the value cannot be less than 32). It returns a rough equivalent in degrees centigrade. The macro is based on the formula, C = 5/9(F – 32):

```
WRITE CENTIGRADE =&
[!UDIVIDE,[!UMULTIPLY,5,[!USUBTRACT,%1%,32]],9]
```

Subtracts 32 from the Fahrenheit argument. Multiplies the difference by 5/9: that is, multiplies the amount by 5, and then divides the product by 9. Finally, outputs this quotient.

UNBLOCK *Command*

Unblock a previously blocked inferior process.

Format

UNBLOCK { username:procname / process-ID } [*username:procname / process-ID*] ...

You must supply the process-ID or the procname. procname can be either a simple or a full process.

The process you want to unblock must be a previously blocked, inferior process unless you have the Superprocess privilege.

Command Switches

/1=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS1 to the specified severity level for this command.
/2=	{ IGNORE / WARNING / ERROR / ABORT }	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) PROCESS  SMITH:PROGA ↓
PID 17
) BLOCK  17 ↓
.
.
.
) UNBLOCK  17 ↓
)
```

First, creates an inferior swappable process that runs concurrently with the CLI. Blocks the new process, performs the required operations, then unblocks the new process.

[!UNE] *Pseudo-Macro*

Include input conditionally.

Format

[!UNE argument$_1$ argument$_2$]

!UNE compares the two arguments. If they are not equal, the CLI executes the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI doesn't execute the input between the !ELSE and the !END.

This pseudo-macro begins a sequence of text that the CLI is to execute conditionally. You must end the sequence with the !END pseudo-macro. The sequence can also include the !ELSE pseudo-macro.

The !UNE pseudo-macro must always have two arguments, which must evaluate to unsigned decimal integers. Both arguments must be in the range 0 to 4,294,967,295.

If the arguments are equal, the CLI does not execute the input up to the !ELSE or !END pseudo-macro. If there is an !ELSE, the CLI executes the input between the !ELSE and the !END.

See Chapter 5 for a discussion of conditional pseudo-macros.

Macroname Switches

None

Argument Switches

None

Example

Given a macro containing:

```
[!UNE,[!VAR0],17]
WRITE VAR0 = [!VAR0]
[!ELSE]
WRITE VAR0 = 17
[!END]
```

If the current value of VAR0 is not 17, this macro writes *VAR0 = n*, where *n* is the current value of VAR0. Otherwise, it writes *VAR0 = 17*.

You can also code the macro as follows:

```
WRITE VAR0 =&
[!UNE,[!VAR0],17][!VAR0][!ELSE]17[!END]
```

[!USERNAME] *Pseudo-Macro*

Expand to the CLI username.

Format

[!USERNAME]

This pseudo-macro does not accept arguments.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE CALL ME [!USERNAME].
CALL ME ISHMAEL.
)
```

[!USUBTRACT] *Pseudo-Macro*

Expand to the difference of two numbers.

Format

[!USUBTRACT argument$_1$ argument$_2$]

The pseudo-macro requires two arguments, which must evaluate to unsigned decimal integers. Both arguments may be double precision. Double precision is in the range 0 to 4,294,967,295.

The value returned is the result of argument$_1$ minus argument$_2$. The pseudo-macro cannot return a negative number. If argument$_1$ is less than argument$_2$, the value returned is equivalent to the absolute value of the actual difference, modulo 4,294,967,296.

Macroname Switches

None

Argument Switches

None

Example

```
) WRITE [!USUBTRACT 17 5]
12
)
```

Subtracts one integer from another and displays the difference.

The macro DIFF.CLI, which contains the following command lines, takes as its arguments two integers. It returns their difference, either positive or negative:

```
[!UGE,%1%,%2%]
WRITE REMAINDER IS [!USUBTRACT,%1%,%2%]
[!ELSE]
WRITE REMAINDER IS -[!USUBTRACT,%2%,%1%]
[!END]
```

First, determines if the first argument is greater than the second. If it is, subtracts the second from the first and displays the result. If the second argument is greater than the first, subtracts the first from the second, puts a negative sign before the difference, and displays the result.

VARn *Command*

Set or display the value of variable VARn.

Format

{VAR0 | VAR1 | VAR2 | VAR3 | VAR4 | VAR5 | VAR6 | VAR7 | VAR8 | VAR9} *[argument]*

This command lets you display the current value of variable VARn or set its current value to the specified value. The CLI provides 10 variables, VAR0 through VAR9. argument must evaluate to a decimal integer in the range 0 to 4,294,967,295.

Command Switches

/1=	{IGNORE | WARNING | ERROR | ABORT}	Set CLASS1 to the specified severity level for this command.
/2=	{IGNORE | WARNING | ERROR | ABORT}	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/P		Set the current value of the variable to the previous environment's value (no arguments allowed).
/Q		Set SQUEEZE to ON for this command.

VARn (continued)

Example

```
) VAR0 ↓
0
) VAR0 34573 ↓
) WRITE [!UADD [!VAR0] 2] ↓
34575
)
```

First, displays the current value of VAR0. Then resets the value. Finally, includes the pseudo-macro !VAR0 in a command line to use the current value of VAR0.

[!VARn] *Pseudo-Macro*

Expand to the current value for VARn.

Format

{ [!VAR0] [!VAR1] [!VAR2] [!VAR3] [!VAR4] [!VAR5] [!VAR6] [!VAR7] [!VAR8] [!VAR9] }

The 10 pseudo-macros !VAR0 through !VAR9 display the current value of the ten CLI variables VAR0 through VAR9. These pseudo-macros do not accept arguments.

Macroname Switches

/P Expand to previous environment's value for VARn.

Argument Switches

None

Example

```
) WRITE THE CURRENT VALUE OF VAR0 IS [!VAR0] ↓
THE CURRENT VALUE OF VAR0 IS 0
) PUSH ↓
) VAR0 39 ↓
) WRITE NOW THE CURRENT VALUE OF VAR0 IS &
&) [!VAR0] ↓
NOW THE CURRENT VALUE OF VAR0 IS 39
) WRITE [!VAR0/P] ↓
0
)
```

First, evaluates [!VAR0] and writes the current value of VAR0. Then changes environment, and gives VAR0 a new value. Evaluates and writes [!VAR0] for the current environment, and then, using the /P switch, for the previous environment.

VSGEN *Utility*

Generate a new AOS/VS operating system (AOS/VS only).

VSGEN is the system generation utility for AOS/VS. It creates an AOS/VS system specifically designed to manage the hardware and software configuration at your installation. VSGEN requests information about the peripheral devices the system will need to manage. It also allows you to adjust system performance by specifying values for certain system parameters.

For a complete description of VSGEN and its operation, consult *How to Generate and Run AOS/VS* (093-000243).

WHO *Command*

Display process information.

Format

WHO [*username:procname* | *process-ID*] ...

This command displays the PID, username, process name, and program name of a process. If you omit the argument(s), the command applies to the CLI.

Command Switches

/1=	IGNORE | WARNING | ERROR | ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE | WARNING | ERROR | ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) WHO ↵
PID: 17 TEDDY CON13 :CLI.PR
)
```

The current process's ID is 17, its username is TEDDY, its process name is CON13, and the program it runs is CLI.PR.

```
) WHO 007 ↵
PID: 7 JAMES_B CON7 :SPY.PR
)
```

WRITE *Command*

Display arguments.

Format

WRITE *[argument]...*

The WRITE command is useful in macros for either writing to the terminal explaining what is happening, or writing a record/log to a LISTFILE explaining what has happened or is happening.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/Q		Set SQUEEZE to ON for this command.

Argument Switches

None

Example

```
) WRITE (A B)_<X Y> )
A_X A_Y
B_X B_Y
)
```

The WRITE command is useful when you want to experiment with the CLI command line operators (parentheses and angle brackets).

You can also use the WRITE command in macros to write to your terminal, explaining what is happening. For example

```
WRITE/L START AT [!DATE] [!TIME]
```

XEQ *Command*

Execute a program.

Format

XEQ pathname *[argument-to-new-program] ...*

The CLI creates a subordinate swappable process with the same priority and privileges that it has. It takes the subordinate process's program from the program file that you specify in pathname. The arguments to the new program are placed in the initial IPC message to the new process. The new program can access these arguments through the ?GTMES system call (see Appendix B for details).

The CLI first tries to execute pathname.PR. If that fails, it tries pathname.

The subordinate process's generic @INPUT, @OUTPUT, and @CONSOLE are the same as its parent's, the CLI's. The subordinate process's generic @LIST and @DATA files are the current list file and data file settings. (Note that these are not necessarily the same as the CLI's generic @LIST and @DATA files.)

The CLI is blocked until the subordinate process terminates. The subordinate's termination IPC message (if any) goes to STRING (if you used the /S switch), to the current list file (if you used the /L switch), the temporary list file (if you used the /L= switch), or to @OUTPUT (if you didn't use either of these switches). The CLI may take exceptional action depending on whether or not the subordinate process returns exceptional condition flags (see Appendix A).

If you are at a console, your console will be available to the program for use as @INPUT. If you are in batch, the program will not have an @INPUT available to it. Using the /I or /M switch causes the CLI to create a special file with contents in accordance with the /I or /M switch. (See HELP topics *I_SWITCH and *M_SWITCH.) This file will be available for @INPUT to the program. The file will be deleted by the CLI when the program terminates. If you run a program requiring the use of @INPUT in batch, you must specify one of these switches on the execute command. Note that a batch input file is not treated as a macro so you will usually want /I, not /M.

Command Switches

/1=	IGNORE / WARNING / ERROR / ABORT	Set CLASS1 to the specified severity level for this command.
/2=	IGNORE / WARNING / ERROR / ABORT	Set CLASS2 to the specified severity level for this command.
/I		Create input for the program from @INPUT. The last line of input must contain a single).
/L		Write CLI output to the current list file instead of to @OUTPUT.
/L=pathname		Write CLI output to the file specified by pathname instead of to @OUTPUT.
/M		Create input for program from the macro body. The last line of the macro body must contain a single).
/Q		Set SQUEEZE to ON for this command.
/S		Return the program's termination message to STRING; default is @OUTPUT.

Argument Switches

Use any argument switches appropriate for the program specified in progname.

Example

```
) XEQ LINK OBJ1 ↵
```

Calls the Link utility; that is, creates a subordinate process whose program is the Link utility. Link produces an executable program file from the object file OBJ1.

```
) XEQ MYPROG 0 1 ↵
```

Executes a program named MYPROG. Note that MYPROG can get the arguments 0 and 1 for use in whatever way you choose.

```
) XEQ/S PROG2 ↵
```

Executes PROG2; diverts PROG2's termination message (if any) to STRING instead of to @OUTPUT.

End of Chapter

Chapter 7
Using Magnetic Tape

This chapter describes using magnetic tape — unlabeled and labeled. It assumes you are acting as a *user*, not doing things like system backups. If you want to do system backups, read about backups in the appropriate *How to Generate and Run* manual for your operating system. To use labeled *diskettes*, see the OPERATOR command in Chapter 6 or the *How to Generate and Run* manual for your operating system.

The major sections in this chapter proceed

- About Unlabeled and Labeled Tape
- Using Unlabeled Tape
- Using Labeled Tape

About Unlabeled and Labeled Tape

Tapes come in different shapes and sizes — from a plastic cartridge with a capacity of 15 Mbytes to a 2400-foot reel with a capacity of 150 Mbytes (at 6250 bpi). The information in this chapter applies to all tapes.

A *label* is information — including a volume ID (volid) — that describes the contents of a tape (i.e., a tape volume). This information is written on the tape by the LABEL utility (when you execute LABEL) and by the operating system (when you or your program issues a command that writes to the tape).

Labels are not mandatory for your mag tape operations — but they do offer several advantages. The main advantages of labeled tape are

- You can store information on multiple tape volumes, without worrying about physical ends of tape. One tape file can span many tape volumes. This is useful for system backup and backup of large database files.
- The tape itself contains information about the material stored on it: user label text, filename, expiration date, and system ID.
- You control the amount of time each tape file will be retained. The system will not overwrite a labeled tape file until the retention period has passed or until the tape is relabeled.
- DG data management programs (INFOS II and DG/DBMS) have logging utilities that expect labeled tapes.
- You can create tapes to be read on other operating systems, like an IBM system or a system that uses ANSI-standard labeled tapes.

If you don't use labeled tape, you'll read and write tape by tape file number. The first tape file is number 0, the second number 1, the third number 2, and so on, up to number 99. You'll be restricted to one tape volume for any read or write operation.

To help you compare labeled and unlabeled tape use, Figure 7-1 shows a sequence of unlabeled and labeled tape operations. These operations dump the contents of a user's directory to tape. The username shown is Jack, but it could be any username.

A tape written without a label (written with a file number) cannot be read as if it were labeled. Generally, the opposite is also true: a tape written with a label and filename can't be read as if it were unlabeled. However, if you don't remember the volid and/or filename on a labeled tape volume, you can display the label (X DISPLAY unitname), note the volid and filename, then proceed with a labeled tape mount operation. You can tell whether or not the tape is labeled with the DISPLAY utility. For example,

```
) X DISPLAY @MTC0:0 )
```

This command displays the contents of the first file of the tape mounted on unit MTC0. On a labeled tape, the first file is the label written by the LABEL utility.

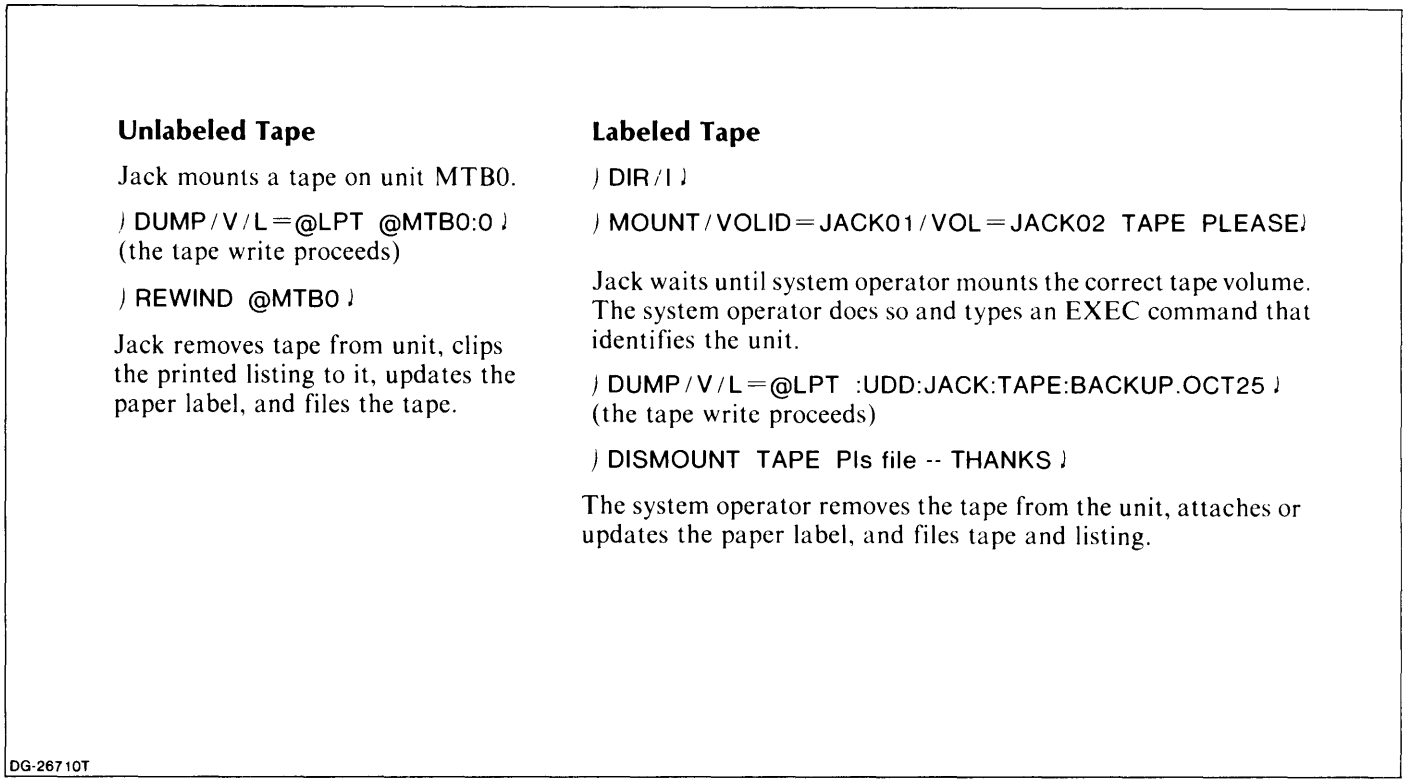


Figure 7-1. Unlabeled and Labeled Tape I/O

Using Unlabeled Tape

There are two kinds of unlabeled tape operations. In the first kind, you choose the unit and mount and dismount the tape yourself; you don't use the MOUNT command, and you act as system operator. For this kind of operation, you need physical access to the tape units. Generally, there is no full-time system operator present. We call this "Unlabeled Tape Operations without the MOUNT Command."

The second type of unlabeled tape operation involves the MOUNT command. It assumes that a system operator is ready and able to physically mount tapes on units. We call this "Unlabeled Tape Operations Using the MOUNT command."

Unlabeled Tape Operations without the MOUNT Command

This kind of operation assumes an *open shop* in which everyone has access to tape units. You must know how to mount tapes, how to write enable tapes, and the unit name and tape filenames. And, you must take responsibility for (paper) labeling and filing of tapes you produce.

This kind of tape operation relies on user expertise and is easy to illustrate. Most tape-oriented examples in this book show this kind of operation.

Examples — Unlabeled Tape Operations without the MOUNT Command

```
) DIR/I ↵
) DUMP/V/L=@LPT @MTB0:0 &↵
&) #\+.ED\+.PR\+.OB\+.ST\?+.TMP+ ↵
```

This dumps all files in and beneath this user's initial directory — excluding files with the suffixes shown — to the first file of the tape mounted on unit MTB0. A listing of all files dumped is printed on the line printer.

```
) DIR/I ↵
) DUMP/V/L=@LPT/AFTER/TLM=[!DATE]& ↵
&) /TYPE=\LNK @MTC0:1 PROJECTS:# ↵
```

This dumps all files in and beneath directory PROJECTS — in this user's initial directory — that were modified today, excluding links. The names of files dumped are listed to the line printer.

```
*) COPY/V @MTC0:0 TBOOT ↵
TBOOT
*) COPY/V @MTC0:1 MY_STANDALONE_& ↵
*&) PROGRAM ↵
MY_STANDALONE_PROGRAM
*)
```

This example shows how to create a tape from which you can bootstrap a stand-alone program (here, called MY_STANDALONE_PROGRAM) that runs without an operating system. To bootstrap, you would put the tape on unit 0 on the first controller, and execute program load steps from device code 22. Then, the system console would ask *FROM MT-0:*. You would type 1⇓; and MY_STANDALONE_PROGRAM would be read into memory and run.

Unlabeled Tape Operations Using the MOUNT Command

The MOUNT command posts a mount request to the system. For the request to be fulfilled, EXEC must be running and a system operator must be on duty. (The command that tells the system an operator is on duty is CONTROL @EXEC OPERATOR ON.) If OPERATOR is not ON, your request receives an *OPERATOR NOT AVAILABLE* message.

Only a user process, running from a console enabled by EXEC, can issue the MOUNT command. Typing a MOUNT command on the system console will produce an error message.

To request an unlabeled tape mount, use the MOUNT command without the /VOLID switch. For example,

```
) MOUNT MYTAPE PLEASE MOUNT A &⇓
&) WRITE-ENABLED 1000' TAPE⇓
```

When it sees a MOUNT command, EXEC creates a link file named for the first argument (here, MYTAPE) in your initial user directory, to be used for tape access.

Also, on the MOUNT command, EXEC displays a mount message on the system console. Someone at the system console must respond to this message. The CLI prompt won't return to your console until someone *at the system console* responds. The response can be to mount the tape and tell EXEC where it is mounted, or to refuse the request. If the operator refuses the request, you'll see a *REFUSED* message. If the operator mounts the tape and tells EXEC where it is mounted, EXEC creates the link file *to the tape unit* in your initial working directory. EXEC also gives your username exclusive access to the unit, so that no other user can accidentally access your tape.

When the prompt returns to your console after the tape is mounted, use the link name to access the tape. The link is always created in your initial working directory, so you must access the tape from this directory or use a pathname to the link (for example, :UDD:JACK:MYTAPE) in all references to the tape. Otherwise, you may get a *DIRECTORY DOES NOT EXIST* or *FILE DOES NOT EXIST* error message.

When you are finished using the tape, you must type a DISMOUNT command using the link name. When you type the DISMOUNT command, EXEC deletes the link from your initial user directory, tells the system to rewind the tape, and prompts the system operator to dismount the tape. When the operator responds to the dismount request, EXEC restores the original tape unit ACL. (This sequence also occurs if you log off before typing DISMOUNT, but typing DISMOUNT is the preferred way to finish up the operation.)

If you interrupt the MOUNT command before the prompt reappears, the link file created by EXEC will remain in your initial user directory. You should type DISMOUNT linkname; this tells EXEC to cancel the request and delete the link it created.

Unlabeled tape MOUNT requests are useful in that you don't need to mount a tape or specify a unit. However, like unlabeled tape operations without MOUNT, they are limited to one tape volume.

Example — Unlabeled Tape Using the MOUNT Command

Assume your username is Chris. A sequence for an unlabeled tape might go as follows.

1. You are logged onto the system under EXEC. On your console, you type

   ```
   ) MOUNT MYTAPE MOUNT A TAPE -- RING IN.⇓
   ```

   This command creates a linkname of MYTAPE, but the name could be any legal filename.

2. EXEC responds to the MOUNT command by displaying a message like this on the *system console:*

   *FROM PID n (EXEC): *** UNIT MOUNT ****
   ...MID=n USER=CHRIS PID=n
   ...REQUEST IS 'MOUNT A TAPE -- RING IN.'
   ...RESPOND: CONTROL @EXEC ....

   If the system operator does not want to mount a tape, he/she types CONTROL @EXEC REFUSED⇓ at the system console, and you get a *REQUEST REFUSED* message. The operator might also send a message explaining the refusal. If the operator does want to mount a tape, he/she gets a blank tape with ring in, physically mounts it on a free unit (say MTC0), and types CONTROL @EXEC MOUNTED @MTC0.⇓

3. The CLI prompt now returns to your console. You can get to the tape by linkname; e.g., MYTAPE:0 for the first file, MYTAPE:1 for the second file. In a program, you can open and read or write to the tape just like any other file. When you are done with the tape, type the DISMOUNT command. For example:

) DISMOUNT MYTAPE STORE AS& ↵
   &)□CHRIS_01. THANKS. ↵

4. EXEC responds to the DISMOUNT command by deleting the link it created, and displaying the following on the system console:

   *FROM PID n : (EXEC) ** UNIT DISMOUNT **
   ...UNIT(S) ARE @MTC0*
   ...REQUEST IS 'STORE AS CHRIS_01.
   THANKS'
   ...RESPOND CONTROL @EXEC
   DISMOUNTED

   (If you forgetfully logged off without typing the DISMOUNT command, EXEC would have sent a similar message to the system console.)

5. The system operator now removes your tape from the unit. After dismounting the tape, the operator types CONTROL @EXEC DISMOUNTED ↵ to tell EXEC the unit is free. EXEC restores the original tape unit ACL.

Using Labeled Tape

As mentioned above, labeled tape has many advantages. Because it can span multiple volumes, it's ideal for system backups and backing up of large database files.

A *label* is information — including a volume ID (volid) — that describes the contents of a tape volume. Some of this information is written on the tape by the LABEL utility when you execute LABEL. Additional label information is written by the operating system, when you or your program issue a command that writes to the tape.

Components of Labeled Tape

There are several different kinds of labels written to a labeled tape. The only ones you really *need* are the volume header label (contains the volid and is created by the LABEL utility), and the first file header label (HDR1, created by the system when it writes material to the tape). Generally, the other labels are useful only if you have application programs that specifically read and write them, via the ?OPEN system call or higher-level language equivalent.

A labeled tape, after it has been labeled via LABEL and after disk-based material has been written to it, has the structure shown in Figure 7-2.

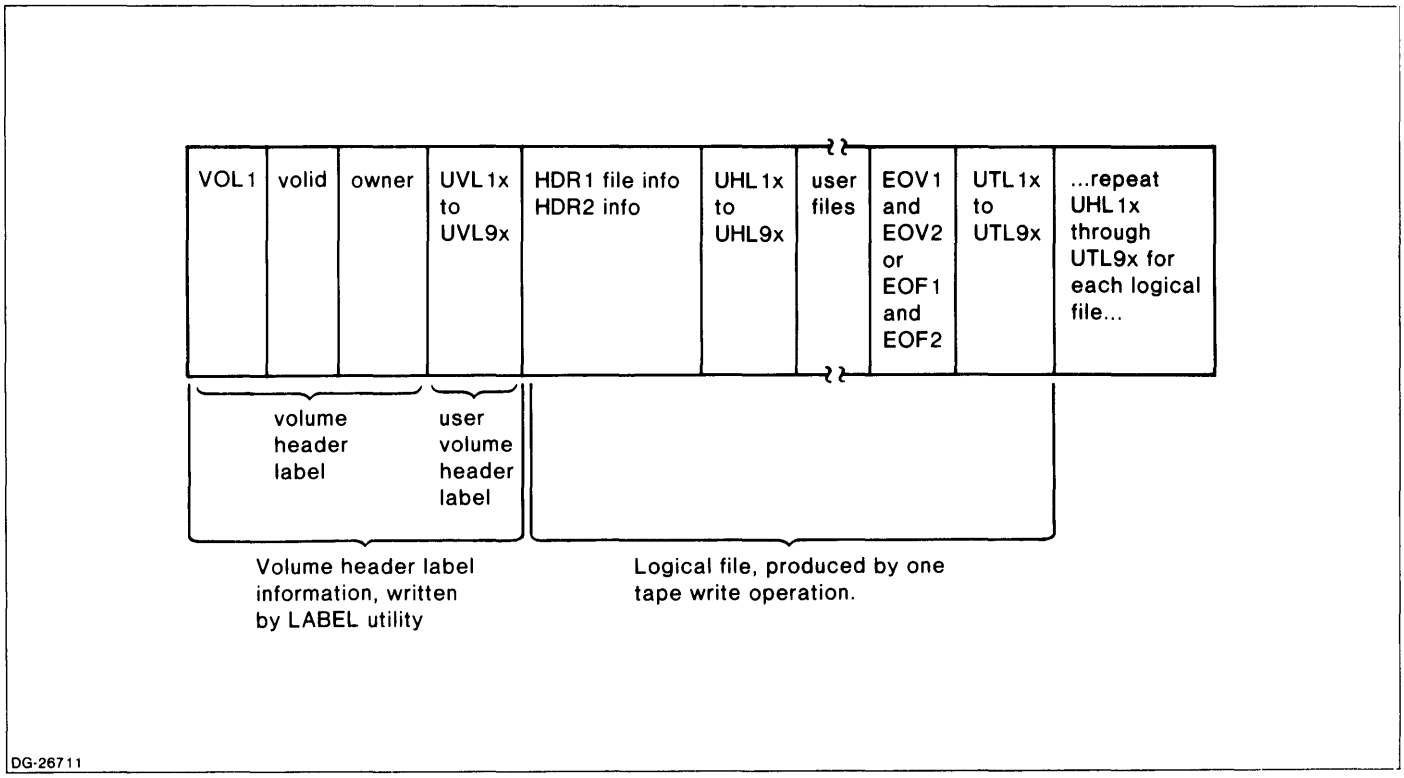


Figure 7-2. Information on a Labeled Tape

The fields shown on the tape have the following meanings.

VOL1 is a fixed string, meaning "volume type 1". The LABEL utility always writes VOL1 on every tape.

volid is the volume ID that you assign using the LABEL utility. The reserved length is 6 characters; the volid can't exceed 6 characters.

owner is the owner field, that you can specify with the LABEL /OWNER= switch. This is optional. The reserved length is 14 characters.

UVL1x to UVL9x are user volume labels you can specify with the LABEL /UVL=x switch. On the tape, each UVL field begins with the characters UVLx, followed by the text of x given with the /UVL=x switch. These labels are optional. Generally, they are useful only if your application programs can read them. The maximum length you can specify for a user volume label is 76 characters.

HDR1 file info HDR1 is written by the system during the write operation. It contains the following items. (All items are created and used only by the operating system, unless noted otherwise.)

- filename for the fileset, as supplied by the person who started the tape write. (This fileset has no relationship to the individual files actually written to the tape.) For example, the command

  ```
  DUMP/V TAPE:XFILE ↵
  ```

  creates the filename XFILE. The maximum filename length is 17 characters;
- fileset ID;
- file section number;
- file sequence number;
- generation number and version number;
- creation date;
- expiration date (the default is 90 days from the creation date; the DUMP command's /RETAIN switch can override the default);
- block count;
- operating system ID (as set by SYSID command, or the default).

HDR2 info This is written to tape (by the system) only if a user program opened the tape using an extended packet on the ?OPEN system call. HDR2 contains the record format (default is fixed length), block length, and record length.

UHL1x to UHL9x are user header labels. These are optional. The only way to have them written is via the ?OPEN system call or higher-level language equivalent. They are useful only if you have application programs that read and write them.

user files This is the disk-based information copied in the tape write. It includes all information written by any *single* CLI command or program write statement. For example, if the command is DUMP, the *user files* will include all files dumped.

EOV1 and EOV2 At the physical end of tape, if there is more material to be written, the system writes a label with EOV1 and EOV2 (an end of volume label). At the end of the last logical file written, the system writes a label with EOF1 and EOF2; this marks the logical end of the labeled tape write operation on this tape fileset.

UTL1x to UTL9x are optional user trailer labels, written after the EOV or EOF label.

DG, ANSI, or IBM Format?

By default, labeled tapes are written in DG format. To write labeled tapes that will be read on a DG, ANSI, or IBM (EBCDIC) system, proceed as follows.

- If the destination system is a DG system (AOS, AOS/VS, etc.), execute the LABEL program without the /I switch. To write to the tape, you can use any CLI command or the default format in any write statement.
- If the destination system is an IBM system (EBCDIC), use LABEL with the /I switch. To write to the tape, you cannot use CLI commands. You must use a program that either opens the labeled tape (?OPEN call to @LMT:volid:filename for IBM format, and specifies field translation in ?WRITE calls); or uses a higher-level language equivalent of ?OPEN/?WRITE to create EBCDIC format.

- If the destination system is an ANSI-based, non-DG system, use LABEL without the /I switch. To write to the tape, you cannot use CLI commands. You must use a program that either opens the labeled tape (?OPEN call to @LMT:volid:filename) for ANSI format; or uses a higher-level language equivalent of ?OPEN/?WRITE to specify ANSI format.

Labeled Tape Management

Generally, tape labeling must be consistent to be worthwhile. There's a big difference between typing

```
DUMP/V @MTB0:2 +.F77 ↵
```

and

```
DUMP/V MYTAPE:F77_SOURCES +.F77 ↵
```

The first approach is hardware-oriented; you must know where to find tapes, how to mount them, and which unit and tape file number to use. The second approach is software-oriented; you can use a real filename of up to 17 characters instead of the unit and file number.

For labeled tape, the tape filenames and labels should be planned, usually by someone acting as system manager or someone in authority. Individual users may not be allowed to create their own filenames and labels, because this would require the system manager to keep track of all the different user choices. If users *are* allowed to do their own filenames and labels, there may be guidelines for them — check this with the system manager if you have any doubts. At some sites, system operators keep a stockpile of labeled tapes for users.

All tape volumes should have paper labels attached to them. The paper label should show the owner, date last written, sequence number, and volid. The tape cover band (for a tape reel) should also have the volid (and perhaps the filename) written on it. The band information identifies each tape as it hangs in the library.

After a labeled tape has been written to, the system will not allow the material on it to be overwritten until the retention period (default 90 days) has passed. The DUMP command and DUMP_II utility have a /RETAIN= switch that can select a different number of days. For example, if a user types

```
) DUMP/V/RETAIN=14 & ↵
&) TAPE:F77_SOURCES +.F77 ↵
```

the system won't allow a labeled tape write to this tape until 14 days have passed. However, anyone can overwrite the *label* (with new retention period and other information) by labeling the tape with the LABEL utility. When you label or relabel a tape, all information recorded on it is lost.

Labeled Tape Operations Using the MOUNT Command

This section describes labeled tape I/O started with the MOUNT command. This is also called an *explicit* labeled tape mount. You can request a labeled tape mount without the MOUNT command (an implicit mount), described later on.

The MOUNT command posts a mount request to the system. For the request to be fulfilled, EXEC must be running and a system operator must be on duty. (The command that tells the system an operator is on duty is CONTROL @EXEC OPERATOR ON.) If OPERATOR is not ON, you will receive an *OPERATOR NOT AVAILABLE* message.

Only a user process, running from a console enabled by EXEC, or in batch, can issue the MOUNT command. Typing a MOUNT command on the system console will produce an error message.

To request a labeled tape mount, use the MOUNT command with one or more /VOLID switches. For example,

```
) MOUNT/VOLID=V1/VOLID=V2 MYTAPE &↵
&) PLS MOUNT A WRITE-ENABLED 1000' TAPE ↵
```

When it sees a MOUNT command, EXEC creates a link file named for the first argument (here MYTAPE) in your initial user user directory, for future access to the tape.

Also, on the MOUNT command, EXEC displays a mount message on the system console. Someone at the system console must respond to this message. The CLI prompt won't return to your console until someone *at the system console* responds. The response can be to mount the tape and tell EXEC where it is mounted, or to refuse the request. If the operator refuses the request, you'll see a *REFUSED* message. If the operator mounts the tape and tells EXEC where it is mounted, EXEC creates the link file *to the tape unit* in your initial working directory. EXEC also gives your username exclusive access to the unit, so that no other user can accidentally access your tape.

When the prompt returns to your console after the tape is mounted, use the link name to access the tape. The link is always created in your initial working directory, so you must access the tape from this directory or use a pathname to the link (for example, :UDD:JACK:MYTAPE) in all references to the tape. Otherwise, you may get a *DIRECTORY DOES NOT EXIST* or *FILE DOES NOT EXIST* error message.

When you are finished using the tape, you must type a DISMOUNT command using the linkname. When you type the DISMOUNT command, EXEC deletes the link from your initial user directory, tells the system to rewind the tape, and prompts the system operator to dismount the tape. When the operator responds to the dismount request, EXEC restores the original tape unit ACL. (This sequence also occurs if you log off before typing DISMOUNT, but typing DISMOUNT is the preferred way to finish up the operation.)

If you interrupt the MOUNT command before the prompt reappears, the link file created by EXEC will remain in your initial user directory. You should type DISMOUNT linkname ⤶; this tells EXEC to cancel the request and delete the link it created.

Volume ID (Volid) Lists

In the MOUNT command, you specify a list of tape volume IDs (volids) with /VOLID= switches. (If you omit the /VOLID switch from the MOUNT command, you specify an unlabeled tape operation.)

On a write (like a dump), you can specify enough volume IDs to hold the disk-based material, or you can use the MOUNT /EXTEND switch to extend the volid list, or both. If the amount of material to be written won't fit on the volid(s) specified, and the /EXTEND switch was omitted, the write will be aborted. Then, you must restart the write from the beginning.

To avoid this situation, you can be generous with the volid list (the system will use only the volids it needs), and/or use the /EXTEND switch. If the latter, EXEC will ask the operator to mount additional tapes as they are needed.

For example, assume you type

```
) MOUNT/VOLID=VOL1/VOLID=VOL2/EXTEND & ⤶
&) XTAPE EXTEND TO VOL3 IF NEEDED ⤶
```

The system operator gets EXEC's mount prompt, mounts a tape with volid VOL1, and types CONTROL @EXEC MOUNTED @MTB0 ⤶.

You then type

```
) DUMP/V/L=@LPT & ⤶
&) :UDD:[!USERNAME]:XTAPE:FILES # ⤶
```

The dump proceeds through VOL1. The system then rewinds the tape and EXEC prompts the system operator for VOL2. The operator mounts a tape with volid VOL2 and types the CONTROL @EXEC MOUNTED command. The dump continues through VOL2. If the dump completes on VOL2, it's done and the /EXTEND switch wasn't needed. But let's assume that at the end of VOL2 more material remains to be dumped. EXEC displays on the system console a message prompting for another volume.

The system operator can then mount a blank tape and label it by using the form X LABEL @MTBn VOL3 ⤶ (as you asked). The operator then types CX MOUNTED @MTBn VOL3 ⤶. The dump then proceeds on VOL3. If needed, the dump can continue through other volumes. When all material you specified has been dumped, the dump is done. You then type DISMOUNT XTAPE ⤶.

MOUNT/EXTEND has the advantage of allowing the operator to extend tape writes (up to the 128-character volid list limit).

Example — Labeled Tape with MOUNT Command

Assume that you are logged on under the username DATABASE. An example in which you write to a multiple-volume fileset might go like the following sequence. (Actually, you would probably do the dump — step 6 — in batch, preceding the DUMP command with QBATCH, to avoid tying up your terminal. Aside from the QBATCH command, all steps shown here would be similar for batch.)

1. Generally, the first step is to discover the volids to specify for the labeled tape operation. This procedure varies. The following shows one way to do it.

   On your console, you type

   ```
   ) SEND 2 I WANT TO DO A DUMP. WILL & ⤶
   &) NEED THREE 2400' BLANK TAPES. RING& ⤶
   &) IN. & ⤶
   &) TELL ME THE VOLIDS TO USE. ⤶
   ```

2. The system console then displays the message you typed:

   FROM PID n: I WANT TO DO A DUMP. WILL NEED THREE 2400 ' BLANK TAPES, RING IN. TELL ME THE VOLIDS TO USE.

   The system operator can then find three blank, labeled tapes or label them now. Then he/she sends you the volid names via the SEND command. For example, you might see

   FROM PID n (OP) USE VOLIDS USR056 USR057 USR058

3. Using the volid information, you type

   ```
   ) MOUNT/VOLID=USR056/VOL=USR057 & ⤶
   &) /VOL=USR058 MYTAPE RINGS IN & ⤶
   &) AND HIGH DENSITY. ⤶
   ```

   This posts a MOUNT request for the three tape volumes, accessible by the linkname MYTAPE (MYTAPE could be any valid filename.)

4. EXEC responds to the MOUNT command and displays the following on the system console.

   *FROM PID n (EXEC): ** UNIT MOUNT **
   ...MID=n USER=DATABASE PID=n
   ...VOLID(S) ARE:
   ...USR056, USR057, USR058
   ...REQUEST IS 'RINGS IN AND HIGH DENSITY.'
   ...RESPOND: CONTROL @EXEC MOUNTED @UNIT...*
   ...

5. The system operator mounts volid USR056 on a free unit (say MTB0) and identifies the unit to EXEC. For example, he/she types CONTROL @EXEC MOUNTED @MTB0. ↵

6. The CLI prompt now returns to your console and you can start your tape write. You could write to separate tape filenames, but for simplicity let's say you decide to use one tape file, called DATABASES. You type

) DUMP/V/BUFFER=8192/RETAIN=14& ↵
   &) /L=DLIST & ↵
   &) :UDD:[!USERNAME]:MYTAPE:DATABASES & ↵
   &) +DATAB+:# ↵

   (Just to be safe, you use a full pathname to the linkname MYTAPE.)

   Your command dumps all disk files in the working directory whose names include the characters DATAB, including all files in any directories whose names include DATAB. The command dumps the files to file DATABASES on the tape. The big buffer size (8192) consumes less tape. The number of days to retain the tape file is 14. The listing file, in the working directory, is DLIST.

7. The tape I/O proceeds. When it reaches the end-of-tape mark on volume USR056, EXEC starts rewinding the tape and prompts the operator to mount the next volume, here USR057.

   The operator removes the tape and mounts the next volume, USR057. He/she then notifies EXEC of the tape unit as above, using a CONTROL @EXEC MOUNTED command.

   Before the I/O proceeds on any volume, the system checks and verifies its volid. If correct, the tape I/O proceeds on volume USR057. (If the volid is incorrect, EXEC displays a "wrong volume" message on the system console.) At the end-of-tape mark on volume USR057, EXEC prompts the operator for the next volume, USR058.

   Once again, the operator dismounts the used volume and mounts the next volume, USR058. He/she then notifies EXEC of the tape unit by typing a CONTROL @EXEC MOUNTED command.

   Again the system checks the new volume's volid. If it is correct, I/O proceeds to the third volume.

8. Somewhere in the middle of this tape volume, the dump is done. All material you specified has been copied to tape. The CLI prompt returns to your console. (Your console would have been tied up by the DUMP command. This is why you might want to do multivolume dumps in batch.)

   You ask the operator to dismount the tape:

) DISMOUNT TAPE THANKS ↵

9. EXEC now prompts the operator to dismount the tape volume and type CONTROL @EXEC DISMOUNTED ↵. After the operator does so, EXEC restores the tape unit ACL to its original setting.

   The operator should file the three tape volumes wherever this type of labeled tape is filed. The tapes will remain read-only until 14 days have passed. (They can be relabeled with the LABEL utility, if desired, but this will destroy all information on the tapes.)

   If you want to reload the file on the three volumes, you could generally follow the procedure (steps 1 through 8). If, when you reloaded the tapes — your username weren't DATABASE, you would need to have Superuser on. Your MOUNT command would specify volids USR056, USR057, and USR058. You could use any valid filename as a linkname. And you would type LOAD/BUFFER=8192/RECENT... instead of DUMP/BUFFER=8192...

   Any load recreates the entire directory/file structure, exactly as it was dumped from the working directory. This can lead to duplication and user confusion. Therefore, be sure to be in the directory from which the dump originated before loading a dump file that contains a directory structure.

Implicit Mount Requests

To help with labeled tape operations, the system maintains a file named @LMT in the peripherals directory. While any labeled tape is mounted, EXEC maintains the user's linkname to file @LMT:current-volid. For example, during DATABASE's dump above, the name TAPE would be linked to file @LMT:USR056, then to file @LMT:USR057, then to file @LMT:USR058.

Even when a labeled tape is *not* mounted, users can use pathname @LMT:volid to request the operator to mount a labeled tape. This type of request is called an implicit mount request.

For example, assume all tape units are idle. You type

```
) LOAD/V @LMT:VOL022:MY_SOURCE.F77 )
```

As with an explicit MOUNT command, the CLI prompt does not return to your console until the system operator takes action.

After your command that includes @LMT, EXEC takes notice and prompts the operator to mount the tape with the specified volid (here, VOL022). The operator can mount the tape and type a MOUNTED command to EXEC, or he/she can refuse the request.

You can specify only one volume (one volid) in an implicit mount request. If more volumes are needed (on a write), EXEC will prompt the operator for them as if you had typed a MOUNT/EXTEND command. Generally, if you or your programs might use an implicit mount, make sure the operator knows about it, so he/she can have the correct volume(s) ready.

Specific Volume Requests

All material written by write operation (the DUMP or COPY command) makes up a single logical file. To read any part of this logical file, the system must read sequentially from the beginning — through multiple tape volumes if needed — until it arrives at the desired part.

There *is* a way to add a logical file to the end of a fileset, or to read from a logical file that starts on a tape other than the first tape in the dump fileset — without reading all preceding tape volumes. This technique, which involves the /SPECIFIC switch for LOAD or DUMP, is useful only if your site does multiple writes (like multiple dumps) to the same tape fileset. For more information on this, see the *How to Generate and Run* manual for your operating system.

User Programs and Labeled Tapes

To gain access to a labeled tape, a program opens file linkname:filename (if MOUNT was typed and the tape mounted before the program was run), or it opens file @LMT:volid:filename (for an implicit mount). In either case, the program must open the tape file for *either* input or output, not both (this is true for all tape I/O). The program must write or read sequential fixed- or variable-length records; it cannot write or read data-sensitive or dynamic records.

If you want to create your own labels rather than use the LABEL utility, you should use the standard formats described in your system Programmer's manual.

Generally, the mount/dismount procedure will be smoother if you request an explicit mount with the MOUNT command before you run programs that need the volume(s). Implicit mounts can be tricky in such situations.

Using Labeled Tapes in Batch

Batch is ideal for large labeled tape dumps and loads. Such batch requests free the user console for other work and they can be queued for a time when workload is light. However, labeled tape I/O *does* require that an operator be on duty — which anyone can ensure by using the /OPERATOR switch with the QBATCH or QSUBMIT command. Such jobs will not start if an operator is not on duty.

Macros to set up and do large-scale dumps in batch are shown in the *How to Generate and Run* manual for your operating system.

Mount/Dismount Summary and Pointers

- The CLI command MOUNT requests the system operator to mount a tape. The MOUNT command must be issued from a user console or a batch stream; it won't work from the system console. A system operator must be on duty. The CLI prompt does not return to your console until the system operator mounts the tape or refuses the request.
- To specify a labeled tape mount, include one or more /VOLID switches in the MOUNT command.
- You can use the LABEL utility to create tape labels, from any console, at any time — even when EXEC is prompting for a tape mount on the system console.
- Within any labeled tape fileset, writes must occur sequentially, from one volume to the next. If you set up multiple filesets with different linknames (e.g., TAPE1, TAPE2, TAPE3), then you can dump to each fileset simultaneously — saving time if you have multiple tape units. If you use only one fileset, you can dump to only one volume at a time.
- Any user can specify *more* volumes than needed for a tape write; e.g., a dump. When all specified material has been written to tape, the CLI command DISMOUNT tells EXEC to ask the operator to dismount the tape. The fileset write will be complete and you can ignore the extra volids specified.

  Or, you can apply the /EXTEND switch to the MOUNT command. If your tape write needs more tape volumes than you specified with /VOLID, the operator will be prompted to mount another labeled tape volume.

  But if you specify *too few* volumes and omit the /EXTEND switch, tape file space will be exhausted

But if you specify *too few* volumes and omit the /EXTEND switch, tape file space will be exhausted before the write is done. EXEC will display a *NO MORE VOLIDS...* error message on your console; and the fileset will be incomplete. The write must be redone from the beginning. Generally, you should be generous with volid lists, or you should use MOUNT/EXTEND switch, or both.

- A 2400-foot tape, at high density (1600 bpi), dumped with a buffer size of 8192 bytes, can hold about 39 Mbytes of disk-based information. At 6250 bpi, a 2400-foot tape can hold about 150 Mbytes. A 1000-foot tape can hold about 16 Mbytes. Either tape can hold slightly more with a bigger buffer size. On an MTB tape unit, a 2400-foot tape takes about 20 minutes to fill (DUMP command) or 12 minutes to fill (DUMP_II program). On an MTC tape unit, a 1000-foot tape reel may take about the same amount of time (12 or 20 minutes) or more. Tape cartridges (15 Mbytes) take more time than reels.
- Batch processing is ideal for multivolume labeled tape I/O.
- An essential part of using labeled tapes is planning and implementing the labels and organizing the tape library.

End of Chapter

Appendix A
Error Messages

This appendix describes the error messages you might see when using the CLI.

Each utility's manual describes the errors that occur while you're running that utility. When you receive an error while running one of your own programs, it usually results from faulty logic in the program. (Invalid syntax evokes an assembler or compiler error). Runtime errors in a program always return an error code in AC0, which the CLI attempts to interpret. For more on such errors, consult the AOS or AOS/VS programmer's manual or the manual for your compiler.

Table A-1. Error Messages Reported Through the CLI

Message	Meaning/Action
NOTHING (NO VISIBLE RESPONSE AT TERMINAL)	Check power, the ON/OFF switch behind the screen, and the ON LINE light. Type CTRL-Q (to undo any CTRL-S that suspended display), or type CTRL-O to undo any CTRL-O.
@^^**&^^^ (TEXT STREAM AND BEEPS)	You may have told the system to type a binary file on your terminal. Try pressing the BREAK (or BREAK/ESC) key, then CTRL-S, CTRL-C CTRL-A and CTRL-Q. If this doesn't work, try BREAK CTRL-C CTRL-B.
ABNORMAL SHUTDOWN	This appears on the system console only. This means that an abnormal AOS or AOS/VS shutdown has occurred. Consult the *How to Generate and Run AOS*, *How to Generate and Run AOS/VS*, *Learning to Use Your Advanced Operating System (AOS)*, or *Learning to Use Your AOS/VS System* manual.
ABORT: text	A utility program hit a fatal error condition and can't continue. If the message allows you to correct the problem, do so. Otherwise, try to find the message that follows ABORT in this appendix.
/AFTER OR /BEFORE SWITCH REQUIRED	When you use either the /TLA= or /TLM= switch, you must also use the /AFTER or /BEFORE switch.
ALREADY BEING PROCESSED	You tried to hold this printing job via a CLI QHOLD command, but the job was already being printed. Use QCANCEL.
AN UNLABELED DISKETTE HAS BEEN INSERTED DO YOU WANT TO (RE)LABEL THIS DISKETTE [N]	The diskette you inserted in the drive is not labeled. If you want to label it, type Y ⟩. The CLI then asks the label name, displaying a default based on the label you specified for the first diskette. If you thought the diskette *was* labeled, or think you may have made a mistake and inserted a diskette with valuable data, press ⟩. Then remove the diskette and find and insert the diskette with the label you expected.

(continues)

Table A-1. Error Messages Reported Through the CLI

Message	Meaning/Action
ARGUMENT IS NOT A COMMAND	You've used an invalid argument in a PROMPT command.
ARGUMENT IS NOT A UNIQUE COMMAND ABBREVIATION	You have entered an invalid abbreviation as an argument to the PROMPT command.
ARGUMENT MAY NOT BE A COMMAND REQUIRING ARGUMENT(S)	An argument in the PROMPT command requires arguments and, therefore, it is an invalid argument to PROMPT.
ARGUMENT MAY NOT BE A NON-IMPLEMENTED COMMAND	You have entered an argument in the PROMPT command not yet implemented.
ARGUMENT MAY NOT HAVE SWITCHES	An argument to the PROMPT command has switches and is, therefore, invalid.
ATTEMPT TO ACCESS PROCESS NOT IN HIERARCHY	Either the process doesn't exist, or your command can't be executed because your process is not the father of the target process.
BINARY MODE NOT ALLOWED	The system operator must enable binary mode with an EXEC BINARY command before anyone can QPRINT/BINARY.
BINARY MODE NOT ENABLED	The system operator has not enabled binary mode on the pertinent printer.
CALLER NOT PRIVILEGED FOR THIS ACTION	The command requires a privilege or process ID that you lack (for example, the Superuser privilege). This message may mean that the command is restricted to the master CLI.
CANCELLED BY OPERATOR	The system operator cancelled your request.
CANCELLED BY USER	You cancelled your request using the CLI QCANCEL command.
CANNOT DELETE UNEXPIRED FILE ON LMT	From the system, on a write command to labeled tape or diskette. The retention period for the fileset (default 90 days, can be selected with DUMP's /RETAIN=switch), has not expired. The system cannot write to the tape/diskette set. You can either select another tape or diskette set for the dump, or use this set. To use this set, you'll need to relabel each tape or diskette. To relabel tapes, use the LABEL utility. Then retry the write. To relabel diskettes, the easiest course is to turn automatic labeling on, using commands) OPERATOR OFF ↵) OPERATOR/LABEL ON ↵ This allows the CLI to relabel the diskettes, overwriting the old labels. Now, restart the dump. For a dump to either labeled tape or diskette, you might consider shortening the retention period via the /RETAIN= switch, so this doesn't happen again.
CANNOT RESTART, /NORESTART SPECIFIED	The system (as always) tried to restart your batch or printing request; or the operator tried to restart your request. But, you specified /NORESTART when you queued the job, so it will not be restarted.

(continued)

Table A-1. Error Messages Reported Through the CLI

Message	Meaning/Action
CAN'T INITIALIZE LD, RUN FIXUP OVER IT	The disk or diskette was not not released properly. The FIXUP program must be run to release it.
CAN'T POP FROM LEVEL 0	You've issued the POP command from LEVEL 0. Level 0 is the highest level.
CHECKSUM ERROR	The unit hardware couldn't read the diskette or tape. Retry. If the error recurs, try another diskette or tape, or different unit (if possible).
CLI OPERATOR ALREADY EXISTS	You tried to turn the CLI operator mode on when it was already on.
COMMAND ABBREVIATION NOT UNIQUE	
COMMAND DOES NOT ACCEPT ARGUMENTS	
COMMAND NOT IMPLEMENTED	
COMMAND REQUIRES ARGUMENT(S)	You have not supplied arguments to a command that requires arguments.
CONFLICTING SWITCHES	One or more command switches contradict each other.
CONSOLE DISABLED FROM LOGGING ON	The system operator has disabled this terminal for EXEC-supervised user log on. You can't log on to it.
CONSOLE INTERRUPT	You've interrupted the process that controls the console with a CTRL-C CTRL-A sequence.
CONSOLE INTERRUPT TASK STACK OVERFLOW	This is a system error; see your system manager.
CONTACT YOUR SYSTEM MANAGER	This usually means EXEC or the system found an error in your user profile. See the system operator or manager.
CONTROL POINT DIRECTORY MAX SIZE EXCEEDED, FILE file	This message means that your command cannot be completed because the current directory is full.
COULDN'T ACCESS CODE FOR MESSAGE	The :ERMES file, which has text for all error codes, is invalid or missing, or it does not include the error code.
DATA GENERAL AOS... text	This is the beginning of the default system banner — appearing on batch output listings and consoles that are ready for user log on. The banner (DATA GENERAL AOS...) can be changed by someone in authority via the CLI command SYSID.
DELETE ACCESS DENIED	You do not have delete (Write) privileges to the requested source file. The delete will not be executed without the deletion.
DEVICE ALREADY IN USE	The diskette or disk is open (has already been initialized). You can release it (**RELEASE ldu-name** ↓) if you want.
DIRECTORY ACCESS DENIED, file	You do not have the Read/Execute access privileges to this directory file.
DIRECTORY DELETE ERROR	The directory you tried to delete has subordinate directories or files.
DIRECTORY IN USE -- CANNOT DELETE	You tried to delete a directory that is in use. The directory could be someone's initial directory, or it could be a directory in someone's searchlist, or the directory is currently in it.

(continued)

Table A-1. Error Messages Reported Through the CLI

Message	Meaning/Action
ENTER YOUR NEW PASSWORD:	Appears on terminal after you type your password but press the ERASE PAGE key or CTRL-L instead of NEW LINE. Type in your new password, 3 to 15 alphanumeric characters. If the password is valid, the system will then say *--NEW PASSWORD IN EFFECT--*
ERROR: message	Find *message* in this appendix for more information.
EXECUTE ACCESS DENIED, file	You do not have the Execute access priviledge to this file.
EXTRANEOUS [!ELSE]	Your macro has a [!ELSE] that is not within a conditional loop.
EXTRANEOUS [!END]	You've typed too many [!END] pseudo-macros in a macro.
FATAL AOS ... ERROR : n *n n n n n* . . (other text) . *STRIKE S FOR ... SHUTDOWN, ...*	Appears on the system console only. A fatal sytem error has occurred. Consult the *How to Generate and Run* manual for your system or *Using AOS on DESKTOP GENERATION Systems* manual.
FILE ACCESS DENIED, file	Your command cannot be executed because you don't have access to the file.
FILE ALREADY EXISTS: file	From system, during load or move command. The file already exists in the specified directory. If you want to move the file into this directory, repeat the command with the /RECENT switch (to keep the most recent version of the file) or the /DELETE switch (to delete the file in the destination directory and replace it regardless of creation date).
FILE DOES NOT EXIST	The system could not find the file you specified.
FILE IS NOT A CONTROL POINT DIRECTORY	You've issued the SPACE command on a file that is not a control point directory.
FILE SPACE EXHAUSTED, file	The tape is full; the dump is invalid.
FIXED RECORD LENGTH IS ZERO	You tried to type a file with zero-length fixed length records.
FLUSHED BY OPERATOR	The system operator has specifically flushed your request.
FORMS ACCESS DENIED	You do not have at least Execute access to directory :UTIL:FORMS and Read access to the file.
FORMS DO NOT EXIST	Directory :UTIL:FORMS does not exist, or the file does not exist.
From PID n: (name) message	This is usually a message from another user.
FROM SYSTEM message	Find *message* in this appendix.
HARD ERROR, DEVICE d u,STATUS = n	This message means that the system cannot read from or write to a tape, disk, or diskette.
ILLEGAL CHARACTER IN PASSWORD *PASSWORD NOT CHANGED*	You typed an illegal characcter in your new password. The legal characters are A–Z (or a–z), 1–9, $ (dollar sign), . (period), _ (underscore), and ? (question mark).
ILLEGAL DECIMAL NUMBER	The argument to this command must be an unsigned, positive, decimal number.

(continued)

Table A-1. Error Messages Reported Through the CLI

Message	Meaning/Action
ILLEGAL DECIMAL SWITCH VALUE	
ILLEGAL FILENAME TEMPLATE	You have entered an invalid filename specification in a command line.
ILLEGAL FILE TYPE, FILE file	The operation you specified is impossible because the file type is wrong.
ILLEGAL FORMAT DUMMY ARGUMENT IN MACRO	You have entered an invalid format for a dummy argument in a macro.
ILLEGAL OCTAL NUMBER	The argument to this command must be an unsigned, positive, octal number.
ILLEGAL REVISION NUMBER	You have entered an invalid argument to a REVISION command.
ILLEGAL SEVERITY LEVEL	The argument to the CLASS1 and CLASS2 commands, and the value of the /1 and /2 command switches, must be IGNORE, WARNING, ERROR, or ABORT.
ILLEGAL VFU CHANNEL AFTER VFU NEXT	A channel number was not between 1 and 12.
ILLEGAL VFU LINE SLEW AFTER VFU NEXT CHAR	Probable cause is a syntax error in the text source file or the FCU-created forms file.
INCORRECT LABELED TAPE FILE	The tape label, or volid sequence, is wrong.
INDECIPHERABLE DUMP FORMAT	The cause may be: 1) This is an unlabeled tape and you tried to treat it as labeled; 2) The file was not DUMPed to the tape (it may have been written with COPY or by a user program); 3) the file was DUMPed with a different buffer size.
INSUFFICIENT MACRO INPUT AVAILABLE FOR /M	You may have omitted the terminating) from the macro input, or the) may not be alone on a line.
INVALID DATE FORMAT	You've entered an invalid argument to the DATE command or to a /TLA= or /TLM= command switch.
INVALID REMOTE USERNAME-PASSWORD PAIR	Your MOVE, COPY, QPRINT, etc., command cannot be executed on the remote system because your password and/or username on the remote system differ from those on the local system.
INVALID TIME FORMAT	You entered an invalid argument to the TIME command or to a /TLA or /TLM command switch.
INVALID USERNAME-PASSWORD PAIR	EXEC cannot find a user profile with this username and password.
LAST MESSAGE CHANGE date time	This describes the time at which :UTIL:LOGON.MESSAGE (whose contents are displayed for each user at log on), was last updated.
LAST PREVIOUS LOGON date time	Specifies the last time someone using your username logged onto this system.
LIST FILE EMPTY, WILL NOT BE PRINTED	You didn't specify output to the batch list file with the /L or /QLIST switches in a batch job.
LOGICAL DISK IN USE, CANNOT RELEASE	The system thinks someone is using the LDU. Check users and get them to log off as needed; then retry the RELEASE command. If the message recurs, check to see if any batch or print job requires files based on the LDU.

(continued)

Table A-1. Error Messages Reported Through the CLI

Message	Meaning/Action
MAY NOT RUN BATCH JOBS	Your user profile does not allow you to submit batch requests.
MESSAGE TOO LONG	The argument string to a SEND command is too long.
MISMATCHED BRACKET TYPES	You've mismatched different types of brackets. One bracket type delimits macros and pseudo-macros; the other expands arguments (see Chapter 3).
MISSING [!END]	You've omitted an [!END] pseudo-macro in a macro.
MUST USE DUMP OR LOAD COMMAND	You receive this message if you use the COPY, MOVE, or WRITE command to read from or write to a labeled diskette.
NEW PASSWORD IN EFFECT	Status message after password change. Your new password is now effective, and the old one ineffective.
NO CLI OPERATOR FOUND	You tried to turn the CLI operator mode off when it was not on.
NO FILES MATCH TEMPLATE	
NO MACRO INPUT AVAILABLE	You've used the /M command switch and there is no terminator in the macro.
NO MORE VOLIDS IN LIST SPECIFIED IN MOUNT COMMAND	File space in your list of volids is exhausted, yet material remains to be written to tape. The labeled tape file is incomplete. One 2400' tape, dumped with high density, 1600 bpi, /BUFFER=8192, holds about 39 megabytes.
NON-UNIQUE QUEUE TYPE ABBREVIATION	
NON VFU CTRL CHARACTER AFTER VFU NEXT	There is an invalid character after ^R.
NO OPERATOR AVAILABLE	Your command (MOUNT or other command with the /OPERATOR switch) requires that a person be available at the system console. Someone must be present at the system console to mount tapes, etc., for this command to work. (See the message below.)
NO OPERATOR AVAILABLE	You tried to use labeled diskettes (using the pathname @LFD:volid:filename), but you have not turned operator mode on for your CLI process. (See the message above.)
NO PREVIOUS VALUE WHEN AT LEVEL 0	You've issued the PREVIOUS command or the /P command switch from LEVEL 0.
NO SUCH QUEUE	The specified queue doesn't exist.
NOT A COMMAND OR MACRO	The text string you typed isn't recognizable as a CLI command or macro.
****NOT ENOUGH MEMORY, RESTARTING CLI****	The CLI required more memory than available to perform your request. DIRECTORY and SEARCHLIST settings remain the same, but the system initializes all other environment parameters (e.g., an extremely large macro file may have attempted execution).
*****NOT ENOUGH MEMORY TO START UP CLI*****	There may be an error in your user profile. See your system manager.
OCCURRED DURING ENVIRONMENT CHANGE	
OCCURRED DURING PROMPT EXECUTION, PROMPT INITIALIZED	An exceptional condition occurred when the CLI executed a command in the PROMPT buffer. PROMPT is set to null.

(continued)

Table A-1. Error Messages Reported Through the CLI

Message	Meaning/Action
PAGE LIMIT EXCEEDED	The system operator enabled page limiting, and the number of pages EXEC estimated for your request exceeds the limit.
PARENTHESES NOT ALLOWED IN MACRO NAME	You've typed parentheses within a macroname, which is illegal.
PASSWORD:	System query message during log on. Type your password followed by ⤵ (NEW LINE).
PASSWORD MUST HAVE 3 TO 15 CHARACTERS *PASSWORD NOT CHANGED*	This may appear after you have tried to change your password.
PATHNAME MUST START FROM WORKING DIRECTORY	You've either specified an invalid pathname or a valid pathname that didn't start at the current working directory in a LOAD, MOVE, or DUMP command.
PATHNAME TOO LONG	Pathnames cannot exceed 128 characters.
PHYSICAL UNIT FAILURE, file	This message means that the system cannot read from or write to a tape, disk, or diskette file.
PHYSICAL UNIT FAILURE *PHYSICAL UNIT OFFLINE*	The disk or tape unit needed is not on line. Use panel switches to put it on line; then retry the command. This message may appear before another error message. Look up the subsequent message in this appendix.
PLEASE INSERT NEXT DISKETTE UNIT [default] *VOLUME ID [default] ? [Y]*	All material on the diskette has been read (on a read); or the diskette is full (on a write). Remove the diskette from the unit. Then insert the next diskette to be read or written to, and press ⤵.
PRIORITY TOO HIGH FOR YOU	Your highest priority (in your user profile) is too low for the queue; or, the priority you gave with the /QPRIORITY switch is higher (closer to 0) than your profile allows.
PROCESS NUMBER TERMINATED BY CONSOLE INTERRUPT	You've aborted the process in control of the console with a CTRL-C CTRL-B sequence.
PROCESS nn TERMINATED [by] ELAPSED TIME: *USER 'user' LOGGED OFF*	The user process that connected your terminal or batch job has been terminated normally, or by operator command, by operator command, or by the system, or by a CTRL-C CTRL-B sequence.
PROFILE NOT FOUND	EXEC could not find a profile with your username.
PSEUDO MACRO ABBREVIATION NOT UNIQUE	The pseudo-macro name you've used is not unique.
PSEUDO MACRO DOES NOT ACCEPT ARGUMENTS	You have supplied arguments in a pseudo-macro that does not accept arguments.
PSEUDO MACRO HAS WRONG NUMBER OF ARGUMENTS	You supplied the wrong number of arguments in a pseudo-macro; one or more arguments may have evaluated to null.
PSEUDO MACRO NOT IMPLEMENTED	You have entered a pseudo-macro not yet implemented.
PSEUDO MACRO REQUIRE(S) ARGUMENTS	You have not supplied arguments in a pseudo-macro that requires arguments.
PSEUDO MACRO UNKNOWN	You've entered an invalid pseudo-macro.
QUEUE NAME UNKNOWN	
QUEUE TYPE UNKNOWN	
READ ACCESS DENIED FILE, file	You lack the Read privilege.

(continued)

Table A-1. Error Messages Reported Through the CLI

Message	Meaning/Action
REQUEST REFUSED BY SYSTEM OPERATOR	The operator specifically REFUSED your request. There may also be an explanation of why the operator did this.
RESTARTED BY OPERATOR	The operator specifically restarted this request.
SEARCH LIST TOO LONG	You've specified a search list which is too long. The search list cannot exceed 511 characters.
SKIP TO VFU CHANNEL N GIVEN CHANNEL N NOT PUNCHED	Syntax error in text source file or forms file.
SOFT ERROR, DEVICE d u, STATUS = o	This appears on the system console. A read or write operation to diskette, disk, or tape failed, then succeeded after the system retried it. If the error occurred on a disk, contact your system manager.
SWITCH ABBREVIATION NOT UNIQUE	You've abbreviated a switch with a nonunique abbreviation.
SWITCH DOES NOT ACCEPT A VALUE	You supplied a value to a switch that does not accept values.
SWITCH REQUIRES A VALUE	You have not supplied a value to a switch that requires a value.
SWITCH UNKNOWN	You entered an invalid switch.
SWITCH VALUE FORMAT ERROR	You've entered an improperly formatted switch.
TAB SENT BEYOND LAST TAB STOP	Syntax error in text source file or forms file.
TERMINATED BY ERROR	An error terminated this process.
THE LABEL ON THIS DISKETTE IS NOT THE LABEL REQUESTED INSERTED: x REQUESTED: y	The diskette you inserted in the drive does not have the right label. On a write (like a dump), it asks if you want to relabel the diskette. On a read (like a load), it asks you to insert it in place of the current one; then press).
THE QUEUE IS FULL	The queue needed by your command was full. The maximum number of entries in a queue is 256.
TOO MANY ATTEMPTS, CONSOLE LOCKING FOR n SECONDS	This appears during a log on attempt. Wait n seconds and try again.
TOO MANY ATTEMPTS, DISCONNECTING	This appears during a log on attempt via a remote terminal. Hang up, redial, and try again.
TOO MANY CHARACTERS IN STRING	The STRING argument exceeds 127 characters.
TOO MANY DIRECTORIES IN SEARCHLIST	A maximum of eight directories may be in a searchlist.
TOO MANY REMOTE FILES OPEN	
TOO SLOW - DISCONNECTING	This appears during a log on attempt via remote terminal. Hang up,redial, and try again.
TOO SLOW - INPUT TIMED OUT	This appears during a log on attempt. If you still want to log on, start over.
UNABLE TO ACCESS QLIST FILE	
UNABLE TO ACCESS QOUTPUT FILE	
UNABLE TO CREATE BATCH INPUT FILE	The CLI cannot create the batch job file because it would exceed the maximum size of the control point or you do not have write access to the directory you are in.

(continued)

Table A-1. Error Messages Reported Through the CLI

Message	Meaning/Action
UNABLE TO CREATE YOUR PROCESS	EXEC couldn't create a user process for you. Your user profile may need to be recreated. See the system operator or manager.
UNKNOWN MESSAGE CODE n	This message means the program cannot find text in file :ERMES for the error code. File :ERMES needs rebuilding.
UNMATCHED [(or <	You've not accompanied the opening [, (, or < with a closing],), or >.
UNMATCHED]) or >	You've not accompanied a closing],), or > with an opening [, (, or <.
USERNAME:	Query message from system; appears during logon. Type your username and ↓ (NEW LINE).
USER NOT PRIVILEGED TO CHANGE PASSWORD	This appears after you try to change your password. Your user profile does not allow you to change your password.
USER SPECIFIED FORM DOES NOT MATCH FORM IN PRINTER	Often this error occurs when the CEO system and EXEC have different settings for printer lines per page.
USER SPECIFIED @OUTPUT ERROR	With the /QOUTPUT switch, you specified an illegal @OUTPUT file.
UTILITY TASK STACK OVERFLOW	This is a system error; see your system manager.
VALID ONLY FROM OPERATOR	Your username is not OP and you issued a CONTROL @EXEC command. Only a process with username OP can give EXEC commands.
WARNING: message	Try to find *message* in this appendix.
WAS NOT MOUNTED	This appears after issuing a DISMOUNT command and the tape linkname you specified was not mounted.
WRITE ACCESS DENIED, FILE file	Your command cannot be executed because you don't have write access to the file.
WRONG NUMBER OF ARGUMENTS	You've supplied the wrong number of arguments to a command or pseudo-macro.
YOU MAY NOT LOG ON FROM A CONSOLE	This appears during a logon attempt. Your user profile does not allow you to use a console. You are restricted to batch.
ZERO DIVISOR	You've supplied a zero divisor to the [!UDIVIDE] pseudo-macro.
ZERO LENGTH FILENAME	Your program or command line specified a file that has zero length.

(concluded)

End of Appendix

Appendix B
CLI/Program Interface

Assembly language programmers can write programs that interface with the CLI in the following two ways:

- If a subordinate process's father is the CLI, it can use the ?GTMES system call to access the command line used to invoke the program. The CLI sends an initial IPC message to each new process. This message contains an edited version of the original CLI command and the command tree. The ?GTMES call provides a convenient way to examine and retrieve portions of this CLI message. (This call can also retrieve an IPC message sent by a father process other than the CLI to its son.)
- If a subordinate process's father is the CLI, it can send the CLI an error code and a termination message, and direct the CLI to enter its abort procedure via the ?RETURN system call.

?GTMES (Get a CLI Message)

Format

?GTMES,*[packet address]*
exception return
normal return

Input

AC2	address of message packet

Output

AC0 and AC1	contents depend upon request type
AC2	unchanged

The system signals exceptional conditions by control taking the exception return with an appropriate condition code in AC0:

FILE SYSTEM codes
IPC codes
MISCELLANEOUS codes

You must supply the system with a parameter packet for the ?GTMES call. Table B-1 describes this packet.

Table B-1. ?GTMES Parameter Packet

Offset	Contents
?GREQ	Table B-2 shows request types.
?GNUM	Argument number. Argument number zero indicates the program name when the CLI format returns.
?GSW	Byte pointer to the name of a simple or keyword switch (used for ?GTSW type calls only). This switch can be no longer than 32 bytes (including trailing nulls) and must not contain any lowercase characters.
?GRES	Byte pointer to the area which will receive the result; −1 if no such area exists. A null always terminates the returned string.

Argument numbers in offset ?GNUM range from 0 (the program name) to n where n is the number of arguments in the command line. The string that offset ?GSW points to should not include the slash and must end with a nul character; i.e., the name of the /L switch is L<0>. If ?GRES specifies no receiving area, then the ?GTMES call will return information in AC0 and AC1 only (see Table B-3).

Offset ?GREQ can contain one of a variety of request types. Table B-2 contains these requests and their names.

If you select ?GCMD, the system reads the edited CLI command into the ?GRES buffer. The original command performs the following editing:

- single commas now separate arguments
- the system removes characters which delimit literals
- the system removes the command, if the command was XEQ, EXECUTE, DEBUG, PROCESS, or CHAIN

All information returned in the ?GRES buffer has a terminating null.

Note that you may use ?GTMES to get messages that are not in CLI format. If you specify no receiving area (−1 in ?GRES), ?GTMES returns flags in AC0 which indicate whether the message is in CLI format or not.

Table B-2. ?GTMES Request Types for Offset ?GREQ

Request	Meaning
?GMES	Copy the entire message to the message buffer specified in offsets ?GRES and ?GREL (unless these offsets contain −1). Return the length of the message (number of words) to AC0, and the message flag, if present, to AC1. The message flag is: ?GFCF Message is in CLI format (the first argument is *program name*).
?GCMD	Read the edited CLI command line into the buffer specified in ?GRES and ?GREL (unless these offsets contain −1). Return the byte length of the edited command line to AC1.
?GCNT	Get the argument count (number of arguments in the CLI command line, excluding the program name) and return this value to AC0.
?GARG	Copy the command line argument specified in ?GNUM, minus switches, to the ?GRES/?GREL buffer area (if that area exists) If the argument consists entirely of decimal digits, the system converts it to binary and returns the results to AC1. (The largest possible value is 65,535.) The system returns the argument's byte length to AC0.
?GTSW	Test the simple or keyword switch referenced by ?GSW to see if it modifies the argument cited in ?GNUM. Return the following test results to AC0: −1 switch not found 0 switch found >0 byte length of the keyword switch If the system finds a keyword switch, it returns that switch's value, terminated by a null, to the ?GRES/?GREL buffer. If the switch value consists entirely of decimal digits, the system converts it to binary and returns the result to AC1. (The largest possible value is 65,535; any larger value causes an error.) If the switch appears in the argument more than once, the system returns information for the first occurrence, only.
?GSWS	Examines the ?GNUM argument for single-character simple switches or keyword alphabetic switches. Set the corresponding low-order bits in AC0 and AC1 for each switch. For example: In AC0 1B0 means /A 1B1 means /B 1B15 means /P In AC1 1B0 means /Q 1B9 means /Z (The remaining bits in AC1 are always 0.)

Table B-3. Parameters Returned by ?GTMES

Request Type	AC0	AC1	?GRES/?GREL Buffer Contents
?GMES	Message Flag (if message is in CLI format)	Total word length of message	Entire message
?GCMD	Unchanged	Byte length of CLI command line	CLI command line
?GCNT	Number of arguments (excluding argument 0, program name)	Unchanged	N/A
?GARG	Byte length of the argument specified in ?GNUM (excluding terminating null)	Binary equivalent of the argument, if argument is decimal; otherwise, −1	Actual argument string
?GTSW	Switch test results: −1 no switch 0 simple switch 1 keyword switch	Binary equivalent of keyword switch, if value is decimal; otherwise, −1	Actual keyword switch
?GSWS	Switch value: 1B0 = /A 1B1 = /B . . . 1B15 = /P	Switch value: 1B0 = /Q 1B1 = /R . . . 1B9 = /Z (Remaining bits = 0)	N/A

?RETURN

The ?RETURN call terminates the calling process and transfers control to its father. The calling process may send a user message to its father if AC1 contains a byte pointer to the message and if the low-order byte of AC2 contains the message length. When the CLI is the process's father, ?RFCF is usually set in AC2 and the message is in CLI command format. If you do not wish to send a user message, set AC2 and its low-order byte to 0. This causes the system to ignore AC1.

If the caller is terminating due to an exceptional condition and if its father is the CLI, the caller may also specify the severity of the exceptional condition by setting high-order byte flags in AC2 and sending an error code in AC0.

Format

```
?RETURN
exception return
```

Input

AC0 Error code to CLI (optional).

AC1 Byte pointer to message (optional).

AC2 High-order byte flags:

?RFCF	Message is in CLI format.
?RFEC	Error code is in AC0; set severity bit:
?RFWA	Warning
?RFER	Error
?RFAB	Abort

Low-order byte:

Length (in bytes) of message.

Severity bits direct the process's father to handle the outstanding exceptional condition as a WARNING, ERROR, or ABORT. This call does not affect the actual severity level (0, 1, 2, or 3) at which the CLI is currently handling WARNINGS, ERRORS, and ABORTS. (See Chapter 4 for a description of severity levels.)

If the process's father is the CLI and the caller issues an exceptional condition flag, the CLI displays the WARNING, ERROR, or ABORT message on the user terminal. If AC0 contains an error code, the CLI displays a system-generated error message on the terminal. If AC1 contains a byte pointer to a user-message, the CLI displays this message on the terminal.

An error return normally sends an error code to AC0. In the example in Figure B-1, we assume that the caller will place an appropriate error code in AC0 before executing the routine.

```
ERTN:   LDA 1, UMLEN      ;LOAD AC1 WITH USER-
                          ;MESSAGE LENGTH.
        LDA 2, FLAGS      ;LOAD AC2 WITH
                          ;EXCEPTION FLAGS.
        ADD 1, 2          ;PUT MESSAGE LENGTH
                          ;IN RIGHT BYTE OF AC2.
        LDA 1, UMPTR      ;PUT MESSAGE BYTE
                          ;POINTER INTO AC1.
        ?RETURN
        JMP ERTN

FLAGS:  ?RFCF+?RFEC+?RFAB ;USER MESSAGE IS IN
                          ;CLI FORMAT, SPECIFY
                          ;EXCEPTIONAL CONDITION
                          ;(?RFEC), AND THE EXCEPT-
                          ;TIONAL CONDITION IS
                          ;"ABORT" (?RFAB).

UMPTR:  .+1*2
UMBEG:  .TXT "PLEASE LEAVE BY THE REAR DOOR"
UMLEN:  .-UMBEG*2         ;USER-MESSAGE BYTE LENGTH
                          ;EQUALS TWICE THE NUMBER
                          ;OF WORDS BETWEEN THIS ADRS
                          ;AND BEGINNING OF MESSAGE.
```

DG-26712

Figure B-1. Sample Error Return Routine

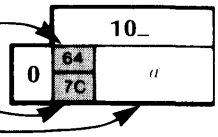
End of Appendix

Appendix C
ASCII Character Set

To find the *octal* value of a character, locate the character, and combine the first two digits at the top of the character's column with the third digit in the far left column

LEGEND:

Character code in decimal
EBCDIC equivalent hexadecimal code
Character



OCTAL	00_	01_	02_	03_	04_	05_	06_	07_
0	0 00 NUL	8 10 BS (BACK-SPACE)	16 10 DLE ↑P	24 18 CAN ↑X	32 40 SPACE	40 4D (	48 F0 0	56 F8 8
1	1 01 SOH ↑A	9 05 HT (TAB)	17 11 DC1 ↑Q	25 19 EM ↑Y	33 5A !	41 5D)	49 F1 1	57 F9 9
2	2 02 STX ↑B	10 15 NL (NEW LINE)	18 12 DC2 ↑R	26 3F SUB ↑Z	34 7F " (QUOTE)	42 5C *	50 F2 2	58 7A :
3	3 03 ETX ↑C	11 0B VT (VERT TAB)	19 13 DC3 ↑S	27 27 ESC (ESCAPE)	35 7B #	43 4E +	51 F3 3	59 5E ;
4	4 37 EOT ↑D	12 0C FF (FORM FEED)	20 3C DC4 ↑T	28 1C FS ↑\	36 5B $	44 6B , (COMMA)	52 F4 4	60 4C <
5	5 2D ENQ ↑E	13 0D RT (RETURN)	21 3D NAK ↑U	29 1D GS ↑]	37 6C %	45 60 -	53 F5 5	61 7E =
6	6 2E ACK ↑F	14 0E SO ↑N	22 32 SYN ↑V	30 1E RS ↑↑	38 50 &	46 4B . (PERIOD)	54 F6 6	62 6E >
7	7 2F BEL ↑G	15 0F SI ↑O	23 26 ETB ↑W	31 1F US ↑←	39 7D ' (APOS)	47 61 /	55 F7 7	63 6F ?

OCTAL	10_	11_	12_	13_	14_	15_	16_	17_
0	64 7C "	72 C8 H	80 D7 P	88 E7 X	96 79 ` (GRAVE)	104 88 h	112 97 p	120 A7 x
1	65 C1 A	73 C9 I	81 D8 Q	89 E8 Y	97 81 a	105 89 i	113 98 q	121 A8 y
2	66 C2 B	74 D1 J	82 D9 R	90 E9 Z	98 82 b	106 91 j	114 99 r	122 A9 z
3	67 C3 C	75 D2 K	83 E2 S	91 8D [	99 83 c	107 92 k	115 A2 s	123 C0 {
4	68 C4 D	76 D3 L	84 E3 T	92 E0 \	100 84 d	108 93 l	116 A3 t	124 4F |
5	69 C5 E	77 D4 M	85 E4 U	93 9D]	101 85 e	109 94 m	117 A4 u	125 D0 }
6	70 C6 F	78 D5 N	86 E5 V	94 5F ↑ or ^	102 86 f	110 95 n	118 A5 v	126 A1 ~ (TILDE)
7	71 C7 G	79 D6 O	87 E6 W	95 6D ← or _	103 87 g	111 96 o	119 A6 w	127 07 DEL (RUBOUT)

SD-00217 Character code in octal at top and left of charts.

↑ means CONTROL

End of Appendix

Appendix D
Submitting Batch Jobs in Stacked Format

When you submit batch jobs to a card reader, they must be in a special *stacked* format. You do not use any CLI commands to submit a job in stacked format, but your job should contain all the CLI commands it needs for processing. Simply place the job in the card reader or give it to the operator.

All stacked batch jobs must have the format illustrated in Figure D-1.

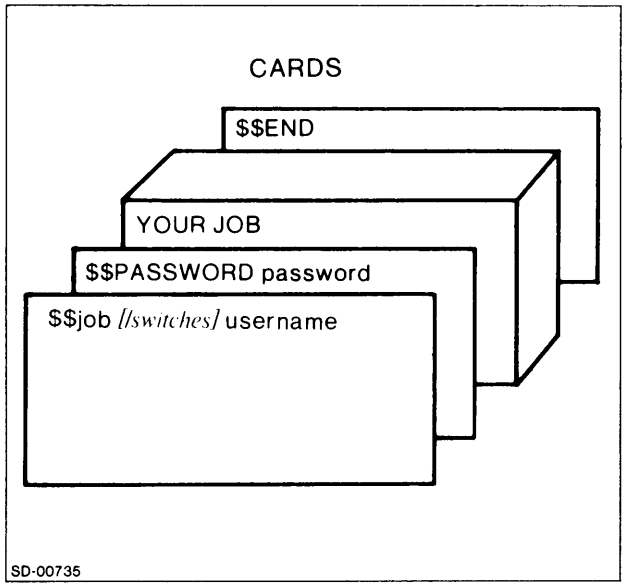


Figure D-1. Stack Format for Batch Jobs

Each job must start with a job control card:

$$JOB *[/switches]* username

Table D-1 contains a list and description of batch job switches. Control cards accept minimum uniqueness; i.e., $$J is equivalent to $$JOB.

Follow the $$JOB card with a password card:

$$PASSWORD yourpassword

For obvious reasons, protect your password card. After the $$PASSWORD card, enter your job. Batch jobs from an actual card reader will contain 80 ASCII characters on a card followed by a NEW LINE character. If a card contains fewer than 80 characters, the reader retains trailing spaces.

The last card in your stacked job must be an end card:

$$END

If your job contains any $$ entries other than those shown above, the system will not queue or process the job.

You must include an END OF FILE card to turn off the card reader. An END OF FILE card has all the holes punched in the first column. For more information, see the *How to Generate and Run AOS* or the *How to Generate and Run AOS/VS*.

Table D-1. Optional Batch Job Switches

Switches	Effect
/CPU=hh*[:mm[:ss]]*	Limit the amount of CPU time this batch job can use. The CLI requires the hours (hh). Minutes (*mm*) and seconds (*ss*) are optional. The maximum legal time is 36:24:32. This switch takes effect only if you have enabled process limiting via the CONTROL @ EXEC LIMITS command.
/AFTER=*[[date:]time]*	Delay processing until a future time. If you specify no date, the system uses today's date. If you specify no time, the system uses midnight. Date is in the form dd-mmm-yy (e.g., 6-MAY-84 for May 6, 1984). You must specify all three parts of the date. Time is in the form hh*[:mm[:ss]]*. The CLI requires hours (hh). Minutes (*mm*) and seconds (*ss*) are optional. You specify the time using a 24-hour system; for example, to begin processing a job after 2 p.m. on August 3, 1984, use the following format: $$JOB/AFTER=3-8-84:14 username
/AFTER=+time	Delay processing until a specific amount of time has elapsed. Time is in the form hh*[:mm[:ss]]*. You can omit the seconds or the minutes and seconds. For example, to begin processing a job in one week, use the following format: $$JOB/AFTER=+168 username
/HOLD	Do not process this job until the user issues a QUNHOLD command to it (by its sequence number).
/JOBNAME	The user must give this job a jobname to issue HOLD and UNHOLD commands to it. (Users can also HOLD and UNHOLD jobs by their sequence numbers.)
/NORESTART	If the system or EXEC fails while it is processing this entry, do not restart the job.
/OPERATOR	This job requires an operator on duty. Use this switch if you need the operator to mount tapes, dismount tapes, etc.
/QPRIORITY=qpriority	Specify a queuing priority for this entry. It must be smaller than or equal to the queuing priority in the user profile. The highest priority is 1; the lowest is 255. If you omit this switch, the default priority is halfway between the limit in the user profile and the minimum (255).
/QUEUE=queuename	Specify the name of the queue to receive this job entry. If you omit this switch, BATCH_INPUT is the default queue.

Between the PASSWORD and END statements, you can issue any CLI command line. The system places these CLI command lines in a disk file, and places an entry for the file on the BATCH_INPUT queue. When job processing begins, the initial CLI environment is as follows:

- The LISTFILE is equivalent to the line printer
- The @INPUT file is equivalent to the file containing CLI command lines
- The DATAFILE is set to NULL CLASS1 is set to ABORT
- All other settings are the normal default values

Figure D-2 illustrates a card file that you can submit to the batch facility. It assembles, binds, and executes an assembly language program, then lists the filenames in the initial working directory. When the batch job completes, the system deletes the file. The system will also delete the file if the batch job is unsuccessful.

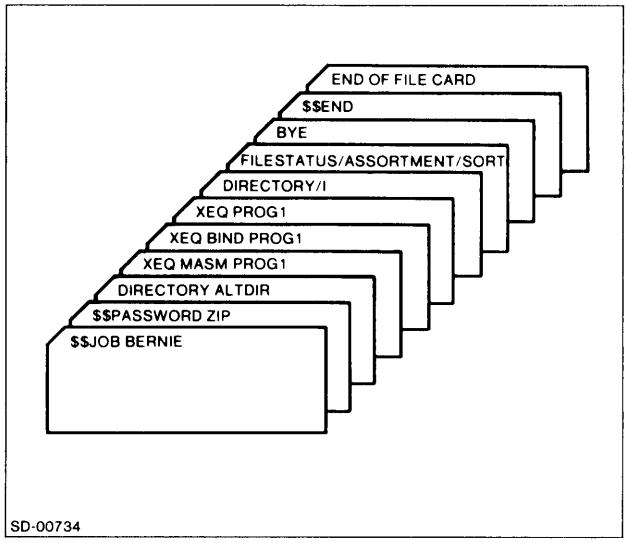


Figure D-2. Sample Batch Job in Stacked Format

End of Appendix

Appendix E
Logging Information into the System Log

You can log accounting information and network error messages into the system log file. Once you've done so, run the AOS Report program to obtain the information. The *How to Generate and Run AOS*, or the *How to Generate and Run AOS/VS* tells you how to start logging and contains a complete description of how to run the Report Program. Here, we supply you with a summary of system logging as you'll apply it to networking. For a more detailed discussion of networking, see the *XODIAC™ Guide for Operators and Network Managers.*

Start System Logging

Issue the CLI SYSLOG command from the *operator's CLI, PID 2*, to start system logging. Note that you can run SYSLOG *only* from PID 2. For example, the command

) SYSLOG/START 06DEC84 ↓

renames the old :SYSLOG log file (if there is any) to 06DEC84 and starts logging in a new file.

If you're running AOS, you'll want X.25 and RMA information in system log file. You must enable ACCOUNTING for both processes. Issue the NETOP commands

) CONTROL @X25 ACCOUNT ↓
) CONTROL @RMA ACCOUNT ↓

to log X.25 and RMA information to the system log file. SVTA does not log information to the system log file.

Executing the Report Program

To execute the Report Program from the AOS CLI, type

) XEQ REPORT/*[switches] [pathname(s)]* ↓

Pathname(s) is the name of the log file(s) you wish to process. If you omit this argument, the system generates log files for all all system log files in your working directory.

You can append the following switches to the REPORT command: /NA, /NE, /FE, /RA, /TA, /FA, /XA, and /XE.

In addition, the /X switch lets you run all XODIAC™ logs on your report.

XODIAC Related Switches for REPORT

/XA or /NA summarizes XODIAC activity reported by the X.25 process. X.25 reports the information *by username*. The information includes the following:

- Link, channel and virtual circuit number (−1 indicates *"NOT APPLICABLE"*)
- Number of virtual connections
- Connect time
- Number of packets transmitted and received
- Number of bytes transmitted and received

/XE or /NE summarizes XODIAC error information reported by X.25. This information includes

- Link, channel, and virtual circuit number (−1 indicates *"NOT APPLICABLE"*)
- Transmit or Receive errors
- Error code number (−1 indicates *"NOT APPLICABLE"*)
- Error diagnostic code in packet

/RA summarizes the XODIAC functional level activity by RMA. RMA reports the information *by username*. The information includes

- RMA connect time
- The number of system calls for remote resources

/FE summarizes XODIAC errors reported by RMA. RMA reports the information *by username* and includes

- HOST ID and virtual circuit number
- Error code number

/TA summarizes the XODIAC functional level activity by FTA. FTA reports the information *by username* and includes

- FTA connect time
- Number of FTA connections
- FTA I/O blocks
- Packets transmitted and received
- Bytes transmitted and received

/FA summarizes the XODIAC functional level activity by FTA and RMA. FTA and RMA report the information *by username* and include

- Total connect time
- Number of FTA connections
- Number of RMA connections
- FTA I/O blocks
- Packets transmitted and received
- Bytes transmitted and received

End of Appendix

Glossary

Abort — The result of a serious error condition. When a program (like the CLI) hits an error, it may display a warning, error, or abort message. The abort message is the most serious of the three: it means the error was so serious that the program couldn't continue.

Access Control List (ACL) — A list of privileges, associated with every directory and file, that specifies the type of access allowed for any user.

ANSI — American National Standards Institute, a committee that publishes standards for a large range of things, including computer languages and tapes, machine screws, and copiers.

AOS — Advanced Operating System, DG's Advanced Operating System for 16-bit ECLIPSE® computers.

AOS/VS — Advanced Operating System/Virtual Storage, DG's Advanced Operating System for 32-bit ECLIPSE computers.

Archive — See backup.

Argument — A value supplied to a command. In general, arguments tell the command what data it should operate with. Consequently, different commands expect different types of arguments; e.g., string or integer. For some commands, the CLI will not accept arguments or the arguments are optional. (See Chapter 6 for a description of all CLI commands and the arguments that they take.)

ASCII — American Standard Code for Information Interchange. The computer industry use the ASCII character set to represent characters as numeric equivalents (See Appendix C).

Backup — The procedure of copying disk-based information for safekeeping, done daily on most computer sites. The copy medium is usually mag tape, but can be disk. Some of the aids to backup are the DUMP command, DUMP_II program (AOS/VS only), and PCOPY program.

Batch — The technique of processing in a continuous, autonomous stream. Batch jobs don't require a console and they run without human interaction (for example, overnight); they are ideal for big, well-defined tasks, like large sorts. You can tell the operating system to run an operation in batch via the QBATCH command.

Baud — The rate at which a line (or modem) can transfer data, in bits per second. Normally, on asynchronous lines, each character requires 10 bits, so characters are transferred at 1/10 the baud rate. The standard (and default) baud rate for consoles is 9600 (960 characters per second). For modems it is 1200.

Block — See disk block.

Blocked Process — A process whose execution is suspended, waiting for an event that may or may not occur. By default, a user process blocks when it creates a (son) process. A process can block voluntarily when it creates a son, via the PROCESS/BLOCK command. Or, another (super) process can block it, or the system can block it.

Boot, Bootstrap — To load a program from a device (tape or disk) into memory and transfer control to it. Bootstrap programs are designed to load other, larger programs like operating systems.

BPI Bits per inch on mag tape; the number of bits stored per inch on each track. There are nine tracks.

Break File AOS —an exact image of a process's address space when the process terminated.

AOS/VS —information on the status of the process at the time of its termination. You can get a break file by typing CTRL-C CTRL-E or by using the CLI BREAKFILE command or by using the CLI command TERMINATE with the /BREAKFILE switch. The system itself will create a break file if certain fatal execution errors occur. (See your system's programmer's manual for more information about break files.)

Break Sequence A control sequence that involves pressing the CMD and BREAK/ESC keys (or BRK or BREAK key). Typed on a user console, the break sequence can break binary mode (used by certain programs when they write to the console), or do other things like log the user off.

Byte 8 bits, can store one ASCII character (e.g., A) or one of 256 different integers.

Central Processing Unit (CPU) The control center of the computer. The CPU performs all arithmetic calculations and directs the activity of peripheral devices.

CEO® System DG's Comprehensive Electronic Office System, which includes electronic mail, calendar, filing, and word processing.

Character Any letter, number, or other symbol represented in eight bits according to the ASCII character set. (See Appendix C for a chart of the ASCII character set.)

CLI (Command Line Interpreter) The AOS and AOS/VS command languages. The CLI commands allow people to communicate with the operating system. When the operating system is brought up, it automatically runs a CLI process as PID 2 on the system console. This is the master CLI, from which all subordinate (son) processes are created. Typically, AOS and AOS/VS run a CLI process for each user who logs on under EXEC.

CMD key A key on the terminal keyboard that, in conjunction with other keys (like BREAK/ESC), can do things like produce the break sequence.

Cold Start Start up when power to the CPU has been off.

Command An instruction to perform a specific sequence of computer operations. Unlike statements, commands execute as soon as you type them. Compiler languages, such as PL/I, consist of statements. In contrast, interpretive languages such as the CLI contain both statements and commands. The CLI commands become statements when they are in macros. See Chapter 3 for a description of CLI command syntax and refer to Chapter 6 for a complete list and description of CLI commands.

Console Any device that allows manual input into the computer and visual and/or audio output from the computer. Display terminals, card punches and readers, and the front panel of the computer (if it has toggle switches and lights) are all examples of consoles. We use the terms console as terminal interchangeably.

Continuation Line A CLI command line that spans more than one line. The maximum length of a line is 128 characters, or 76 characters if SCREENEDIT is ON.

Control Character Any character whose ASCII octal value is between 0 and 37. Enter control characters into the computer by pressing the CTRL key and another key at the same time.

Control Point Directory See directory.

Crash See panic.

Cursor On a CRT console screen, the cursor indicates the current position on a line. Two examples are a box superimposed on a character position and an underscore beneath a character position.

Database A central location (one or more disk files) for data, which can be shared by more than one user or program. DG has three systems for creating and managing database: INFOS II, DG/DBMS, and DG/SQL.

Data File A file that contains data. Data files are distinct from other types of files because you can both enter and extract information. You can create a data file by using the CLI CREATE command with the /I switch, or by using a text editor. To examine the contents of a data file, use the CLI command TYPE or the DISPLAY utility.

Default A value or characteristic that the system assigns unless you specifically direct it to assign a different value.

Delimiter A character, or series of characters, that denotes the end of a command line. The CLI recognizes the following five characters as valid delimiters: the NEW LINE key, the carriage return (CR) key, form feed (CTRL-L), end-of-file (CTRL-D), and null (ASCII 0).

Directory File A special type of file that catalogs information about subordinate files. You cannot store data in directories but you can use them to organize a hierarchical file structure. (See Chapter 2 for more information about directories.)

Directory Tree A file structure represented hierarchically as an inverted tree. (See Figure 2-1 in Chapter 2.)

Disk Block An area on a disk that contains 512 bytes of data.

Disk Formatter A program supplied with the system, in both stand-alone and stand-among versions, used to create LDUs, change LDU specifications, and check disks' surfaces for flaws (bad blocks).

Dummy Argument A variable that represents an actual or potential argument. (See Chapter 5 for a complete description of dummy arguments and their uses in macros.)

Dump To copy information en masse onto another medium. Most dumps are done for backup. Sometimes a dump is done to provide diagnostic information, as in a memory dump (which copies computer memory so programmers can examine it, try to reproduce the problem, and solve it).

Dump_II Program An AOS/VS utility program that copies disk-based material using the same format and switches as the CLI command DUMP. DUMP_II is faster than DUMP.

Echo The output of what was just input. When you type on a keyboard, for instance, the video display or hardcopy terminal echos the characters.

Environment Each user process has a number of attributes (e.g., search list, level, prompt). Collectively, these attributes define the environment of the process. (See Chapter 4 for a complete list and description of the parameters that make up the environment.)

EXEC A utility supplied with the operating system that manages user consoles, batch and spool queues, and user mount requests. EXEC accepts commands from any process with the username OP and sends messages to the system console.

Extension (Suffix) Some system utilities assume that filenames have specific extensions. If you just enter the filename without the extension, the utility will add the needed extension. It can be useful to name your files with their appropriate extensions. The extensions tell you at a glance what kind of file it is. Another name for filename extension can be filename suffix.

Father Process A process that creates another process. Father processes, also called parent processes, are superior to their son processes. A father process can assign privileges to a son process.

FCU (Forms Control Utility) A utility program that inserts control characters in files for special-form printing.

FED A disk file editor utility supplied with AOS/VS.

Field Engineer A DG engineer whose primary responsibility is hardware.

File A collection of data whose only necessary relationship is sharing the same file. Files are essentially organizational tools: they allow you to retrieve groups of information by specifying a single filename. (See Chapter 2 for more information about files.)

Filename The name of a file. Either you or the system gives the file a name when you or the system creates it. You can change the name of a file with the CLI command RENAME. (See Chapter 2 for more information about filenames.)

Fileset A name for one or more disk files written (usually dumped) to a set of one or more labeled mag tape volumes. For example, if a user asks for a three-volume mount and writes one or more disk files to them in a single operation, the fileset includes the three tape volumes. The fileset ID is written to the tape label and allows the system to keep track of the tape volumes that contain the fileset.

FTA The XODIAC network file transfer agent. It helps copy files from one computer system to another.

Function Key Special keys that programs use to perform various operations. The CLI does not use function keys.

Generic Filenames Files that the operating system automatically takes input from, and sends output to, for various commands and utilities. AOS and AOS/VS have these generic files: @CONSOLE, @INPUT, @OUTPUT, @LIST, @DATA, and @NULL. You can set these generic filenames to represent different files. (See Chapter 2 for more information about generic files.)

Hash Frame Size A number that helps determine where the operating system will put the filename of each file it creates in a directory. Each directory has a hash frame size. The default is 7.

Hierarchy (Process) All processes are related in a structure that resembles an inverted tree. The highest processes are the peripheral manager process (PMGR, PID 1) and the master CLI process (PID 2).

Host A computer system that's connected to one or more systems. The system you are on is called the local host; any of the other systems is called a remote host.

I/O Device Any machine that facilitates input to, and output from, the computer.

Input As a verb, the process of entering information into the computer. As a noun, input refers to information which has been, is being, or will be entered into the computer.

Interrupt A request for system service (e.g., termination, I/O request, error handling). Interrupts can be either manual (as when you type CTRL-C CTRL-B) or the system itself can generate an interrupt (as when an error occurs).

Labeled Tape A magnetic tape volume with a label at its beginning. There are industry-wide standards for such tapes, within the broad groups ANSI and IBM. The LABEL utility can write either kind of industry-standard label to a tape. The label information includes the tape volume ID (volid) and other information. Labeled tape has many advantages over unlabeled tape; we recommend it for all of your tape backups.

LABEL Utility Creates ANSI or IBM labels on tapes; also see labeled tape and magnetic tape.

Laser Document Printer A printer that produces high quality copy which resembles typeset copy.

LDU Logical Disk Unit; one or more physical disks, processed by the Disk Formatter into one logical directory file.

Letter-Quality Printer A printer that produces copy suitable for a business letter. The copy resembles typewritten copy.

Level You have a number of potential levels to which you can PUSH or POP. For each level, you can define environment parameters. (See Chapter 4 for a description of levels and environments.)

Line (of Text) A sequence of ASCII characters that ends with either a NEW LINE, carriage return, form feed, or null character.

Link File A special type of file that contains a pathname. When you specify a link filename in a command, the CLI substitutes the contents of the link (a pathname) for the link filename. You can create links by using the CLI command CREATE with the /LINK switch. (See Chapter 2 for more information.)

LOAD_II Program An AOS/VS utility program that restores material dumped from disk by either the DUMP command or DUMP_II program.

Local Area Network (LAN) A network of computer systems that are relatively close to one another — up to a mile apart. This is a good arrangement for one or more MV/Family systems (as hubs of the network), and DESKTOP GENERATION™ systems as remote hosts.

Local (item) An item (like system, console, or printer) that is managed by your computer system without a communications line. The opposite of local is remote. For example, in a XODIAC network, from your local console, you can log on to a remote system and use the remote printer. Or, when you use a modem line, your local console communicates with the remote system.

Logical Disk Unit (LDU) One or more physical disks, processed by the Disk Formatter into a logical unit.

Macro An abbreviation of macroinstruction. A macro is a single instruction that stands for a series of instructions. (See Chapter 5 for more information about CLI macros.)

Magnetic Tape The medium used for file backup and archiving. Types of tape units are MTAn (AOS only) (800 bpi), MTBn (1600 or 800 bpi), MTCn (1600 bpi only, streaming or cartridge) and MTD (AOS/VS only) (6250 or 1600 bpi). Unlabeled tape files are numbered 0, 1, 2, 3,... and so on. Users append the number to the device name to specify the tape file; e.g., @MTB:0. Tapes can be labeled with the LABEL utility. Users specify labeled tape filenames via their own linkname and tape filename; e.g., TAPE:FILE.

Master CLI See CLI.

Modem A device that connects an asynchronous line from a computer to a telephone line, and connects the other end of a telephone line to a console. Two modems are needed for each remote console.

Multiplexor (Mux) A general term for a device that sorts and controls multiple signals. In this book, a multiplexor is part of a user-console handling device called an IAC (Intelligent Asynchronous Controller) or ATI (Asynchronous Terminal Interface).

Operating System The programs that decide what parts of the computer will be running at any given time (resource management), and which code gets executed. Most operating systems also contain a number of system calls. (See your system's programmer's manual for more information.)

Output As a verb, the process of translating information stored in the computer to a form usable by human beings. As a noun, output refers to data generated by computer operations.

Overlay A section of executable program code that can be called into a reserved area of memory as needed, then overwritten by another overlay called into memory. Overlays associated with a program usually reside in an overlay file that has the suffix .OL.

Page In memory, a 2,048-byte quantity.

Page-Seconds Indicates a process's memory usage in relation to time; formed by multiplying the number of memory pages used by the number of CPU seconds used. For AOS (not AOS/VS), page-milliseconds (pages*ms) are used instead of page-seconds. The PED and REPORT utilities both give page-second figures for processes.

Panic An error that halts CPU processing with a fatal error message on the system console.

Parent Process Same as father process.

Patch A correction or update made to a program on disk. DG provides patch files in updates for AOS and AOS/VS programs like EXEC, AOS or AOS/VS systems and the CLI.

Pathname The route you must take to get from your working directory to your destination-file. Pathnames must be relative either to the root directory or to your working directory. (See Chapter 2 for more information about pathnames.)

PCOPY A fast stand-alone or stand-among (AOS/VS only) copy program supplied with the operating system, used to back up an LDU by making a physical copy of it, onto another LDU or onto magnetic tape.

PED (Process Environment DISPLAY) An utility program that can display all processes' vital statistics.

Peripheral Device Any part of the computer system except the central processing unit (CPU). Line printers, card-readers, teletypes, digital plotters, and display terminals are some examples of peripheral devices.

Peripheral Directory (PER) The directory that holds all device entries. Its full pathname is :PER or shorthand prefix @. The prefix @ that you use with devicenames specifies the peripherals directory.

Peripheral Manager A set of programs supplied with the operating system that supervises all character I/O (e.g., with user consoles). The base program always runs as PID 1. The files are all in the root directory.

PID The numeric process identifier that the system creates and associates with each process.

PMGR See peripheral manager.

Preditor The user profile editor supplied with the operating system which creates user profiles that identify system users to EXEC, and allows them to log on and off.

Process One or more tasks that compete as a group for system resources, such as memory, file space, I/O devices, and CPU time. Every process has a unique process identification number (PID) that you can display with the CLI WHO command. (See your system's programmer's manual for more information about processes.)

Program A sequence of computer instructions. Programs can exist in a variety of forms (e.g., written in different programming languages, stored on tape or disk, listed in octal or decimal values). In whatever form, program always refers to a set of computational steps.

Prompt A signal that tells you a program is waiting for input.

Pseudo-Macro All CLI pseudo-macros begin with an exclamation point. The CLI has two types of pseudo-macros: those that expand to a numeric or string value (e.g., !TIME and !DATE) and those that specify conditional execution (e.g., !EQUAL and !ELSE). The CLI uses the first type of macro as an argument or function, whereas the second type acts as a command. (See Chapter 5 for more information about pseudo-macros and how to use them in macros.)

Queue A file that stores print, batch, and other requests until the system is ready to process them.

Record A series of one or more characters written to or read from a file.

Release Notice A notice of recent software changes that DG hasn't yet been able to include in pertinent manuals, supplied with the operating system and other software as a printed listing and disk file in :UTIL (pathname :UTIL:RELEASE:n.nn; e.g., :UTIL:RELEASE.4.00).

Remote (item) An item (like a system, console, or printer) managed by another computer or by your computer over a communications line. The opposite of remote is local.

Revision A new version of software and manuals, or a new version of microcode.

Root Directory The apex of the directory tree. The root directory's name is : (colon), but you should not confuse it with the colon you use to separate directories in a pathname. Whenever you see a colon at the beginning of a pathname, it refers to the root directory. Colons in the middle of a pathname separate filenames.

Search List The list of directories that the CLI searches through when you specify a filename. Every process environment has a search list. You can set, change, or display your search list with the CLI command SEARCHLIST. (See Chapter 2 for more information about searchlists and refer to Chapter 6 for a description of the SEARCHLIST command.)

Separator A character, or sequence of characters, that denotes the end of an expression. The separators for CLI commands are: a space (or series of spaces); a tab (or series of tabs); a comma; or any combination of spaces, tabs and a comma. These characters separate commands from arguments and arguments from other arguments.

The semicolon is called the command separator. You use it when you want to enter two or more commands on the same input line. (See the section on Multiple-Command Input Lines in Chapter 3.)

Software Subscription Service A service that provides new revisions of operating systems and support software as DG creates them.

Software Trouble Report (STR) A formal report, made by a customer to DG through a DG systems engineer, about a problem that the customer is having with the software. The cause may be a user or DG error. DG personnel try to duplicate the problem to solve it, thus they need as much information about the problem as possible. Or, instead of reporting errors, an STR can simply offer suggestions.

Son Process When one process creates another process the first process is the father process and the second process is a son process. A son process is subordinate to its father process.

String A series of characters. For example, ABC is a string.

Subordinate Directory A directory that is relatively lower on the directory tree.

Superior Directory A directory that is relatively higher on the directory tree.

Superprocess A privilege that allows a user process to control any process on the system.

Superuser A privilege that allows a user to bypass file access controls and refer to any file on the system.

Switch A predefined character or character sequence that alters the functioning of a command. The CLI uses two types of switches: those that you append to a command and those that you append to an argument. Some switches accept values while others do not. (See Chapter 3 for more information about switches.)

System Call A routine residing in the operating system's address space. System calls save both time and space — they relieve the programmer of the burden of coding commonly used routines and they provide protection and security for storage devices (e.g., disks, tapes).

System Console The console connected directly to the CPU. User consoles are not connected directly, but through a multiplexor that sorts incoming and outgoing data flow.

System Engineer A DG engineer whose primary expertise is software, secondarily hardware. Field engineers generally handle hardware problems.

Task An atomic unit of activity in a process. The term task usually refers to a program in execution. A single process may have several tasks running within it, with each one getting a piece of the process's CPU time.

Template Character A symbol that represents a set of characters. The CLI recognizes the following templates: + (plus sign), - (dash), # (pound sign), * (asterisk), and \ (backslash). (See Chapter 2 for more information about templates.)

Terminal An interactive device that has a keyboard for input and for display either a screen as on a video display terminal or a printed listing as on a hardcopy terminal.

User Directory The directory created and maintained for each interactive user. Usually becomes the working directory when the user logs on. It can have subordinate directories.

User Directory Directory (:UDD) The system's directory that contains each user directory.

Username The name with which you log on.

User Profile A disk file, created via the PREDITOR utility, that contains each user's username, password, disk space allowance, and other privilege specifications.

UTIL The utilities directory, contains most (if not all) utility programs on the system. UTIL's pathname, :UTIL, is often found in search lists.

Utility (Program) Utilities are programs that Data General supplies. :UTIL contains most utilities.

Virtual Console A device entry on the system that allows remote users to log on as if they were on a local console; provided by DG's XODIAC network VTA (Virtual Transfer Agent). VTA's virtual consoles are unrelated to real consoles.

Volid The six-character volume ID that identifies each labeled tape, written with the LABEL utility. (See Chapter 7.)

Warm Start Startup in which power has remained on to the CPU.

Working Directory The directory in which you currently reside. Even if you access a file that is not in your working directory by using a pathname or your searchlist, your working directory will remain the same. All pathnames should be relative either to the root directory or to your working directory. (See Chapter 2 for more information about your working directory.)

X.25 The XODIAC network management support process, which runs all other network operations. X.25 is the name of an international standard for intercomputer communications.

XODIAC DG's networking system, which allows one DG system to communicate with one or more other DG systems in a private or a local area network. XODIAC also allows DG systems to participate in a Public Data Network like TELENET.

End of Glossary

Index

D

E

M

T

X

Y

Z

Data General

TP_____

TIPS ORDER FORM

Technical Information & Publications Service

BILL TO:
COMPANY NAME_____
ADDRESS _____
CITY _____
STATE_____ ZIP _____
ATTN: _____

SHIP TO: (if different)
COMPANY NAME_____
ADDRESS _____
CITY _____
STATE_____ ZIP _____
ATTN: _____

QTY	MODEL #	DESCRIPTION	UNIT PRICE	LINE DISC	TOTAL PRICE

(Additional items can be included on second order form)

[Minimum order is $50.00]

Tax Exempt #_____
or Sales Tax (if applicable)

TOTAL	
Sales Tax	
Shipping	
TOTAL	

METHOD OF PAYMENT

☐ Check or money order enclosed
For orders less than $100.00

☐ Charge my ☐ Visa ☐ MasterCard
Acc't No._____ Expiration Date_____

☐ Purchase Order Number:_____

SHIP VIA

☐ DGC will select best way (U.P.S or Postal)

☐ Other:
☐ U.P.S. Blue Label
☐ Air Freight
☐ Other _____
_____

NOTE: ORDERS LESS THAN $100, INCLUDE $5.00 FOR SHIPPING AND HANDLING.

Person to contact about this order _____ Phone _____ Extension _____

Mail Orders to:

Data General Corporation
Attn: Educational Services/TIPS F019
4400 Computer Drive
Westboro, MA 01580
Tel. (617) 366-8911 ext. 4032

Buyer's Authorized Signature Date
(agrees to terms & conditions on reverse side)

Title

DGC Sales Representative (If Known) Badge #

DISCOUNTS APPLY TO MAIL ORDERS ONLY

012-1780

educational services

CUT ALONG DOTTED LINE

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE
TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form shown on the reverse hereof which is accepted by DGC.

1. **PRICES**
   Prices for DGC publications will be as stated in the Educational Services Literature Catalog in effect at the time DGC accepts Buyer's order or as specified on an authorized DGC quotation in force at the time of receipt by DGC of the Order Form shown on the reverse hereof. Prices are exclusive of all excise, sales, use or similar taxes and, therefore are subject to an increase equal in amount to any tax DGC may be required to collect or pay on the sale, license or delivery of the materials provided hereunder.

2. **PAYMENT**
   Terms are net cash on or prior to delivery except where satisfactory open account credit is established, in which case terms are net thirty (30) days from date of invoice.

3. **SHIPMENT**
   Shipment will be made F.O.B. Point of Origin. DGC normally ships either by UPS or U.S. Mail or other appropriate method depending upon weight, unless Customer designates a specific method and/or carrier on the Order Form. In any case, DGC assumes no liability with regard to loss, damage or delay during shipment.

4. **TERM**
   Upon execution by Buyer and acceptance by DGC, this agreement shall continue to remain in effect until terminated by either party upon thirty (30) days prior written notice. It is the intent of the parties to leave this Agreement in effect so that all subsequent orders for DGC publications will be governed by the terms and conditions of this Agreement.

5. **CUSTOMER CERTIFICATION**
   Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

6. **DATA AND PROPRIETARY RIGHTS**
   Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

7. **DISCLAIMER OF WARRANTY**
   DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANT ABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS SUPPLIED HEREUNDER.

8. **LIMITATIONS OF LIABILITY**
   IN NO EVENT SHALL DGC BE LIABLE FOR (I) ANY COSTS, DAMAGES OR EXPENSES ARISING OUT OF OR IN CONNECTION WITH ANY CLAIM BY ANY PERSON THAT USE OF THE PUBLICATION OF INFORMATION CONTAINED THEREIN INFRINGES ANY COPYRIGHT OR TRADE SECRET RIGHT OR (II) ANY INCIDENTIAL, SPECIAL, DIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOSS OF DATA, PROGRAMS OR LOST PROFITS.

9. **GENERAL**
   A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer.

DISCOUNT SCHEDULES

DISCOUNTS APPLY TO MAIL ORDERS ONLY.

LINE ITEM DISCOUNT

5-14 manuals of the same part number - 20%
15 or more manuals of the same part number - 30%

DISCOUNTS APPLY TO PRICES SHOWN IN THE CURRENT TIPS CATALOG ONLY.

Data General

TIPS ORDERING PROCEDURE:

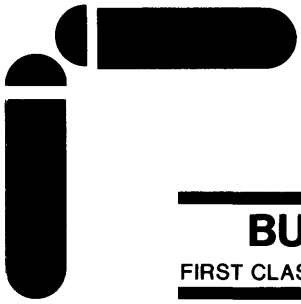
Technical literature may be ordered through the Customer Education Service's Technical Information and Publications Service (TIPS).

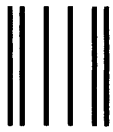
1. Turn to the TIPS Order Form.
2. Fill in the requested information. If you need more space to list the items you are ordering, use an additional form. Transfer the subtotal from any additional sheet to the space marked "subtotal" on the form.
3. Do not forget to include your MAIL ORDER ONLY discount. (See discount schedules on the back of the TIPS Order Form.)
4. Total your order. (MINIMUM ORDER/CHARGE after discounts of $50.00.)

   If your order totals less than 100.00, enclose a certified check or money order for the total (include sales tax, or your tax exempt number, if applicable) plus $5.00 for shipping and handling.
5. Please indicate on the Order Form if you have any special shipping requirements. Unless specified, orders are normally shipped U.P.S.
6. Read carefully the terms and conditions of the TIPS program on the reverse side of the Order Form.
7. Sign on the line provided on the form and enclose with payment. Mail to:

   TIPS
   Educational Services - M.S. F019
   Data General Corporation
   4400 Computer Drive
   Westboro, MA 01580
8. We'll take care of the rest!

educational services





NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA 01772

Postage will be paid by addressee

DataGeneral

Customer Documentation
MS E-219
4400 Computer Drive
Westboro, MA 01581-9973

moisten & seal

CUSTOMER DOCUMENTATION COMMENT FORM

Your Name _____ Your Title _____

Company _____

Street _____

City _____ State _____ Zip _____

We wrote this book for you, and we made certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve the manual. Please take a few minutes to respond. Thank you.

Manual Title _____ Manual No. _____

Who are you?

- ☐ EDP/MIS Manager
- ☐ Analyst/Programmer
- ☐ Other _____
- ☐ Senior Systems Analyst
- ☐ Operator
- _____
- ☐ Engineer
- ☐ End User

How do you use this manual? *(List in order: 1 = Primary Use)*

- ___ Introduction to the product
- ___ Tutorial Text
- ___ Other _____
- ___ Reference
- ___ Operating Guide

fold

About the manual:		Yes	No
	Is it easy to read?	☐	☐
	Is it easy to understand?	☐	☐
	Are the topics logically organized?	☐	☐
	Is the technical information accurate?	☐	☐
	Can you easily find what you want?	☐	☐
	Does it tell you everything you need to know?	☐	☐
	Do the illustrations help you?	☐	☐

If you wish to order manuals, use the enclosed TIPS Order Form (USA only).

Comments:

Data General Corporation, Westboro, MA 01580

093-000122-07