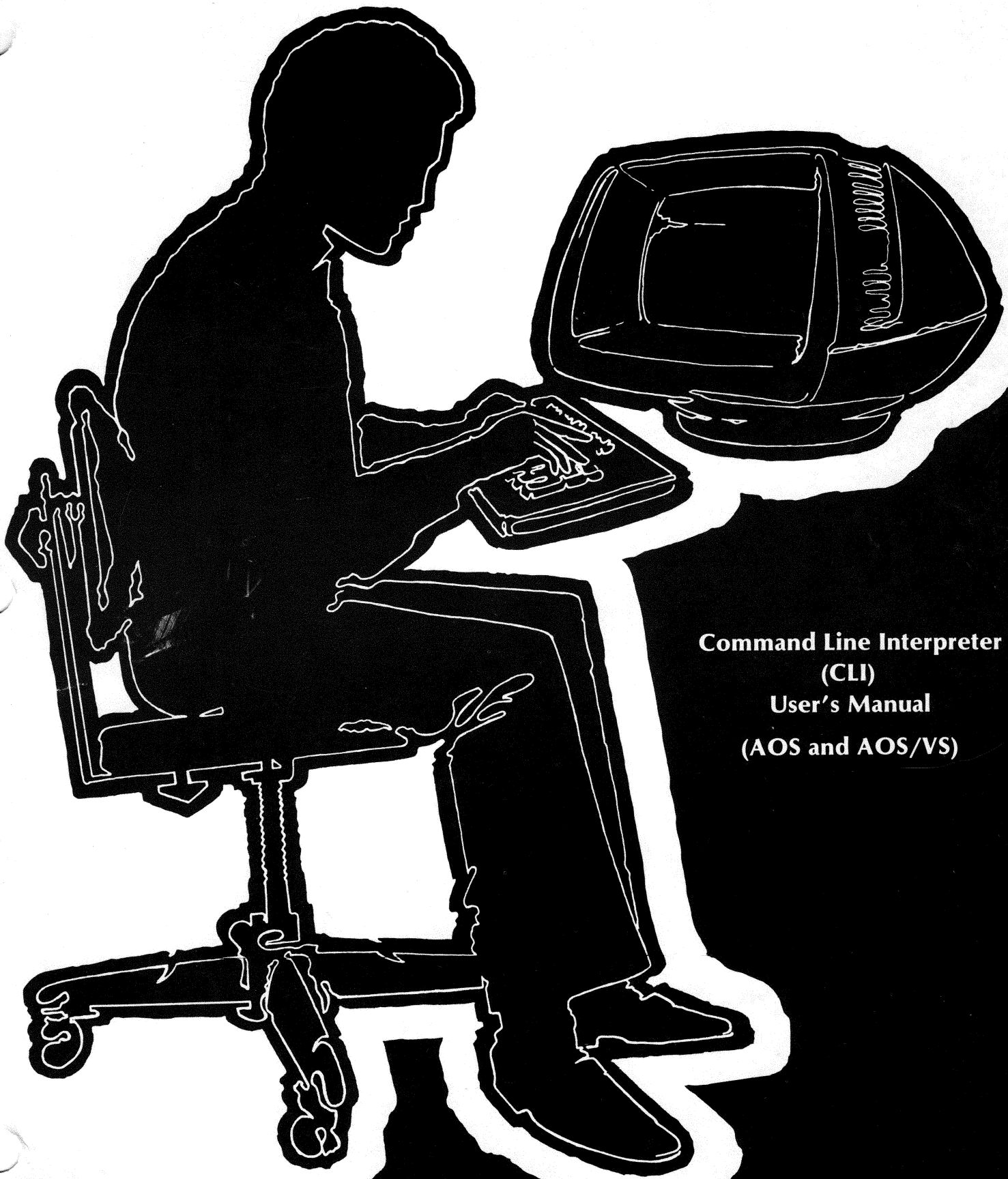
DataGeneral



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

093-000122-05

Command Line Interpreter

(CLI)

User's Manual

(AOS and AOS/VS)

093-000122-05

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 05, May, 1982

NOTICE

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

093-000122-05

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

Effective with:

(AOS Rev. 4.00) (AOS/VS Rev. 1.40)

A vertical bar or an asterisk in the margin of a page indicates substantive change or deletion, respectively, from the previous revision.

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. 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 should assume that we are speaking about both CLIs.

If you have not used AOS or AOS/VS, 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.

We have written this manual as a tutorial text, except for Chapter 6, "CLI Commands, Pseudo-Macros, and System Utilities" which is a reference chapter.

Chapter 1 introduces the CLI and explains console control characters. It also contains a glossary of technical terms which we use throughout the manual.

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 control them.

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

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

Appendix A lists the exception condition 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).

We also supply a tab with this manual. It sets off Chapter 6 to aid speedy access to this important reference chapter.

Suggested Manuals

Many concepts introduced in this manual are 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 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.
- The *AOS/VS Programmer's Manual* (093-000241) describes the AOS/VS operating system in detail. It explains system calls, memory management, file input/output, and multitasking.
- *The AOS Operator's Guide* (093-000194) describes how to run AOS on a routine basis. It explains how to run the EXEC, Spooler, and other system utilities. It also explains how to handle system failures and CPU powerfail.
- The *AOS/VS Operator's Guide* (093-000244) describes operating procedures for AOS/VS.
- *Managing AOS/VS* (093-000243) explains how to load and generate AOS/VS and how to manage the system once it is generated.

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: $\left\{\begin{matrix}\text{required}_1\\ \text{required}_2\end{matrix}\right\}$ 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.)

All numbers are decimal unless we indicate otherwise; e.g., 35_8.

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.
- Require additional manuals -- Please contact your local Data General sales representative.
- Experience software problems -- Please notify your local Data General systems engineer.

End of Preface

Contents

Chapter 4 (continued)

Chapter 5 - CLI Macros

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

Chapter 6 (continued)

Appendix A - Exceptional Condition Messages

Appendix B - CLI/Program Interface

Appendix C - ASCII Character Set

Appendix D - Submitting Batch Jobs in Stacked Format

Appendix E - Logging Information into the System Log

Tables

Illustrations

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. 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 the line printer and tape drive. 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 execute the same command a number of times with different arguments. Additionally, the CLI has a macro capability (see Chapter 5) which allows you to designate a series of commands. Then, when you enter the name of the macro, the CLI executes all of the designated 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 purpose for the CLI. Others, which we describe in the next section, perform special operations.

DELETE

The delete key (DEL), named RUBOUT on some keyboards, allows you to erase characters after you have typed them in. 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 deletes 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. They tell the CLI that you have finished typing on the current line. If you are typing a command, hitting NEW LINE or CR will tell the CLI to begin executing the command. If you are typing something into a file, NEW LINE or CR will move the cursor to the leftmost position of the next line. The only difference between the two terminators is that CR truncates characters to the right of the cursor while NEW LINE does not.

BREAK

The BREAK key puts your terminal back in text mode. If, for instance, you mistakenly type the contents of a 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 put your terminal back in text mode, press the break key followed by CTRL-S, CTRL-C, CTRL-A and CTRL-Q.

ERASE PAGE

On video display terminals, the ERASE PAGE key will clear the screen and move the cursor to the top left-hand corner of the screen. This key also acts as a command terminator.

NOTE: If the page mode characteristic is on, ERASE PAGE may freeze the terminal. Type CTRL-Q to free it.

HOME

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

REPEAT

Use the repeat (REPT) key in conjunction with another key. REPT will keep inputting the other character for as long as you hold the two keys down. Use REPT carefully -- it repeats very fast.

SHIFT

The SHIFT key prints alphabetic characters in uppercase and prints 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 will cause

all subsequent alphabetic characters to be uppercase. Conversely, if all your characters are uppercase, pressing the ALPHA LOCK key will cause 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 is no 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. 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 1-1. Control Characters

Control Character	Effect
CTRL-C	Begin a control character sequence
CTRL-D	End-of-file
CTRL-O	Discard or restart data written to the terminal
CTRL-P	Transmit the next character without interpreting it to the process performing the read operation
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 empty the input buffer if you had typed ahead. Refer to the control character sequence section 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: CTRL-D causes an exception return to the task which performed the read operation. The CLI will terminate if it gets two zero-length end-of-files in a row while reading from its input file. In other words, two CTRL-Ds typed in a row will terminate the CLI.

CTRL-O causes information being output by an executing program to be discarded rather than written 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 when it receives another CTRL-O. (Note the change from previous releases: CTRL-Q no longer restarts output that has been 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 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 which performed the read operation.

CTRL-S stops displaying information on the terminal and suspends the task when it attempts to write 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 by typing CTRL-Q; the system automatically cancels CTRL-S when the current 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 the CLI is in continuation mode, CTRL-U cancels only the current input line, not the entire command line.

Control Character Sequences

You can affect the execution of a process which 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, as shown in 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. If the process has not defined a console-interrupt service task, CTRL-C CTRL-A is ignored. 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. 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. This is not the preferred way to log off; you should use the BYE command instead.

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

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.

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-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. Simply type

```
) HELP name↵
```

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

```
) HELP/V DUMP↵
```

To review a list of CLI topics, type HELP without any arguments.

For more information about HELP, see the HELP command in Chapter 6.

Glossary of Technical Terms

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, arguments are not accepted or are optional. (See Chapter 6 for a description of all CLI commands and the arguments which they take.)

ASCII — American Standard Code for Information Interchange. The ASCII character set (see Appendix C) is used throughout the computer industry to represent characters as numeric equivalents.

Break File — AOS -- an exact image of a process's address space at the moment the process was 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 command TERMINATE with the /BREAKFILE switch. The system itself will create a break file if certain fatal execution errors occur. (See the AOS or AOS/VS programmer's manual for more information about break files.)

Central Processing Unit (CPU) — The control center of the computer. The CPU performs all arithmetic calculations and directs the activity of peripheral devices.

Character — Any letter, number, or other symbol that can be represented in eight bits according to the ASCII character set. (See Appendix C for a chart of the ASCII character set.)

Command — An instruction to perform a specific sequence of computer operations. Unlike statements, commands are executed as soon as you type them. Compiler languages, such as PL/I, consist of statements. In contrast, interpretive languages such as the CLI are made up of both statements and commands. The CLI commands become statements when they are used 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. In general, when we use the term *console*, we mean the front panel of the computer. We will refer to other types of consoles as terminals.

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.

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 a tree. (See Figure 2-1 in Chapter 2.)

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.)

Echo — The output of what has just been input. When you type on a keyboard, for instance, the characters are echoed on your video display or hard-copy terminal.

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 which make up the environment.)

Father Process — A process that has created another process. Father processes, also called parent processes, are said to be superior to their son processes. A father process can assign privileges to a son process.

File — A collection of data whose only necessary relationship is sharing the same file. Files are essentially organizational tools: they allow you to access groups of information by specifying a single filename. (See Chapter 2 for more information about files.)

Filename — The name of a file. Every file is given a name when it is created, either by you or by the system. You can change the name of a file with the CLI command RENAME. (See Chapter 2 for more information about filenames.)

Function Key — Special keys that different programs use to perform various operations.

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.)

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, etc.). 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).

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.)

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.)

Macro — An abbreviation of macroinstruction. A macro is a single instruction which stands for a series of instructions. (See Chapter 5 for more information about CLI macros.)

Operating System — The programs that decide what parts of the computer will be running at any given time (resource management), and which code will be executed. Most operating systems also contain a number of system calls. (See the AOS or AOS/VS 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.

Parent Process — Same as father process.

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.)

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 all peripheral devices.

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) which you can display with the CLI WHO command. (See the AOS or AOS/VS 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 first type of macro is used as an argument or function, whereas the second acts as a command. (See Chapter 5 for more information about pseudo-macros and how to use them in macros.)

Root Directory — The apex of the directory tree. The root directory is named : (colon), but it should not be confused 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.

Son Process — A process created by another process is called the son of the creating 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.

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, etc.).

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 — 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.)

Username — The name with which you log on.

Utility (Program) — Utilities are programs that Data General supplies. Most utilities are located in :UTIL.

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.)

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 describe files, filenames, directories, the directory tree, pathnames, templates, search lists, links, access control lists, generic filenames, and file types.

A *file* is a collection of information. It may reside on any medium, 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 is a string of 1 to 31 characters. The following are 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.

Filename Extensions

Some system utilities assume that filenames have specific extensions. 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.

The following are among the extensions that AOS and AOS/VS system utilities recognize.

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

Directory Files

Although there are many different types of files, in this section we place files into one of two categories: directory files and data files (at the end of this chapter, we'll list all file types). A *directory* is a file that catalogs and contains information used 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, 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.

Directories are organized into a network resembling an inverted tree. We say that a directory is superior to directories on lower branches of the tree. Directories on lower branches are called subordinate directories.

Figure 2-1 shows a directory tree. The top directory in the tree is called the root; since you will refer to it often, we gave it the filename : . Three of its 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 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 merely separates the filenames.

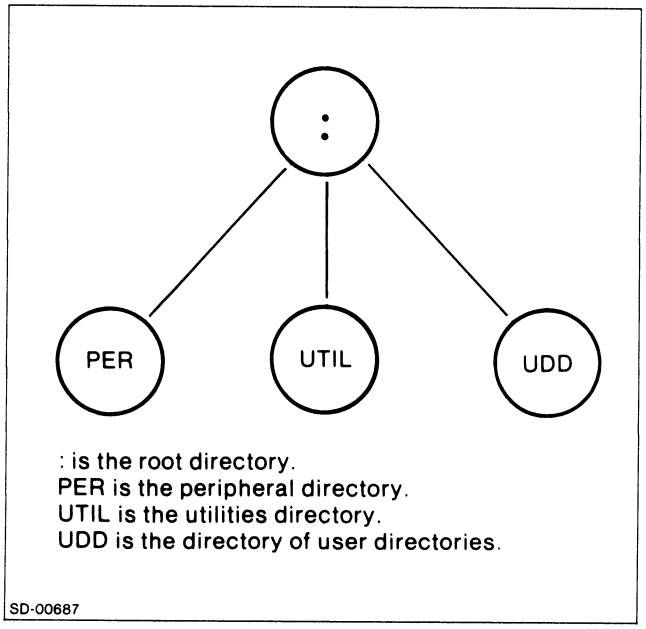


Figure 2-1. A Directory Tree

Every user starts with a directory subordinate to UDD. This *initial working directory* has the same filename as the user's username. 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. 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.

While you are on the system, you use a particular directory as a reference point in the tree. This reference point is called the *working directory*. To refer to any file in your working directory, all you use is the file's name. For example, in Figure 2-2 if your working directory is CAROL and you want 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 furnish a pathname to the file.

Pathnames

To refer to a file outside your working directory, you may use a pathname. A *pathname* describes the route the system takes from directory to directory to get to the file you want. A pathname can be a prefix, one or more filenames separated by colons, or a combination of prefix and filenames. Except for the last filename, all filenames in a directory must be directory names. The general format for a pathname is:

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

This means that each pathname can consist either of a prefix alone, or of an optional prefix followed by one or more filenames, each separated by a colon. (Remember: Directories are themselves a type of file, so directory names are filenames too.)

The pathname prefixes are:

:	(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 current 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 searchlist directories. If you do use the equal sign, the system will not look through your searchlist (see the section on searchlists later in this chapter).

↑	(UPARROW)	Move up to the parent directory. This is the only way to move up the directory tree. You can use more than one uparrow as the prefix for a pathname.
@	(AT SIGN)	Start at the peripheral directory. This is equivalent to :PER:.

Unless you use the colon or @ prefix, your pathname is always relative to your current 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 reference FILE2 with the pathname =UDD:TED:FILE2. This is an illegal pathname since UDD is superior to CAROL; the CLI will return an error. The pathname ↑TED:FILE2, however, will reference FILE2.

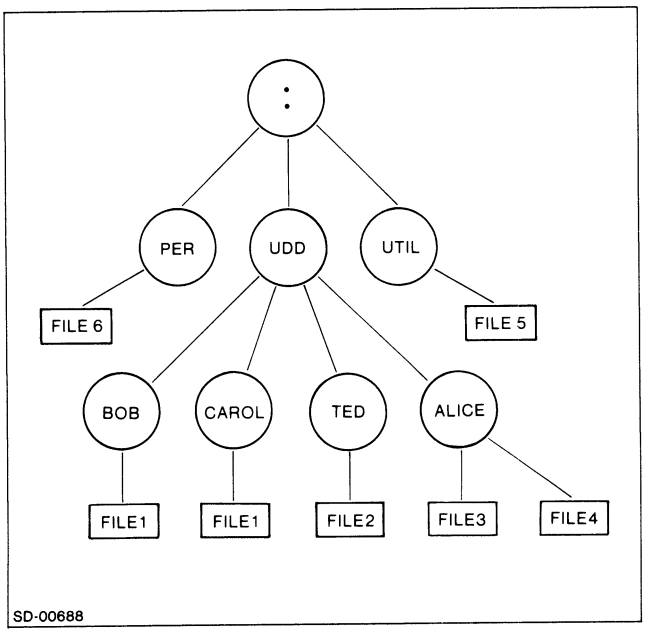


Figure 2-2. Sample Pathnames

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 traverses to find the files they specify.

When you supply a pathname in a CLI command, your working directory doesn't change. (Note that you can always find the name of your working directory by typing the DIRECTORY command.) For example, if in Figure 2-2 the working directory is CAROL, the command

```
) QPRINT ↑↑UTIL:FILE5↵
```

prints FILE5 in directory UTIL, but your working directory remains CAROL.

Templates

A template character is a character that the CLI interprets symbolically rather than literally. A pathname template is a pathname which contains template characters. In a sense, prefixes are template characters since the CLI does not evaluate these characters literally. Prefixes, however, are limited to the beginning of a pathname. The template characters that we describe in this section can occur anywhere in the pathname. Table 2-1 lists the five CLI templates and the filenames that they match.

Table 2-1. Template Characters

Template Character	What it Does
* (asterisk)	Matches any single character except a period
- (hyphen)	Matches any character string that does not contain a period, including a null string
+ (plus sign)	Matches any character string, including strings with a period and null strings
# (pound sign)	Expands to + and +:+ and +:+:+ and so on until all subordinate directories have been matched
\ (backslash)	Tells the CLI to exclude any filenames that match the following template

The number sign and backslash template characters are described more fully in the following sections. The asterisk, hyphen and plus sign are quite easy to use and are very powerful.

Suppose you want to access a file but can't remember the exact filename. You are sure, however, that it begins with *S*. If you type

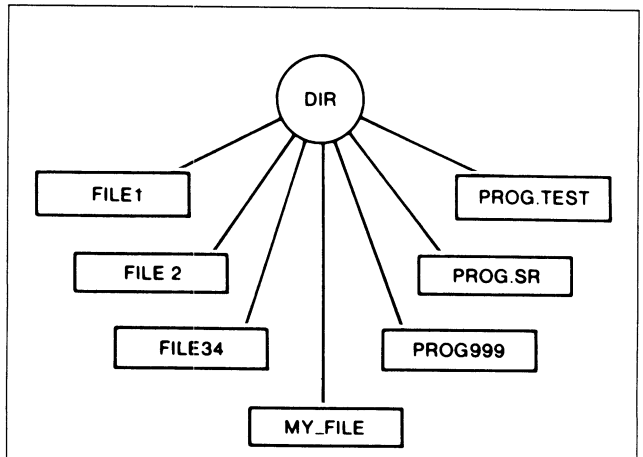
```
) FILESTATUS S+↵
```

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.ST). Rather than issuing a separate MOVE command for each file, you could type

```
) MOVE PROGRAMS PROG1+↵
```

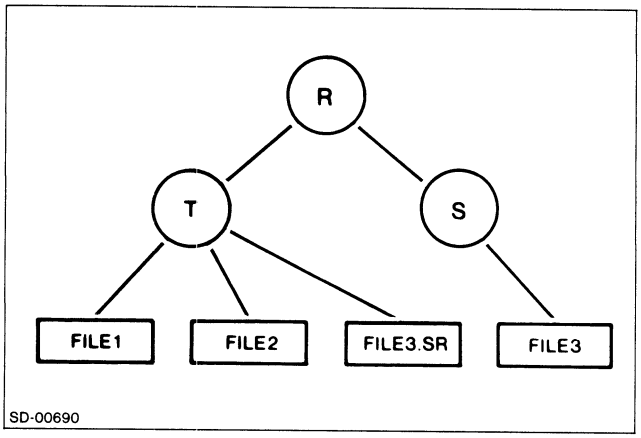
and the CLI will move all files in your working directory that begin with PROG1 to the subordinate directory PROGRAMS. See Figure 2-3 for examples of template matching.



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



SD-00690

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 template matches a LINK file, it will not resolve the link.

Number Sign

Unlike the other template characters, the number sign can expand to lengthy pathnames consisting of more than one filename. The number sign expands to the filename immediately preceding it and all 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

=

and

=+:#

and

=+:+:+#

and so on. The expansion halts automatically when it matches the bottom of the tree (every directory subordinate to the directory immediately preceding the number sign).

This template character is very useful for finding a file when you don't know which directory it's under. 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 exactly where it's located. If you type

```
) FILESTATUS E↓
```

the CLI will return another prompt telling you that it didn't find E under the current working directory. If, however. you type

```
) FILESTATUS #:E↓
```

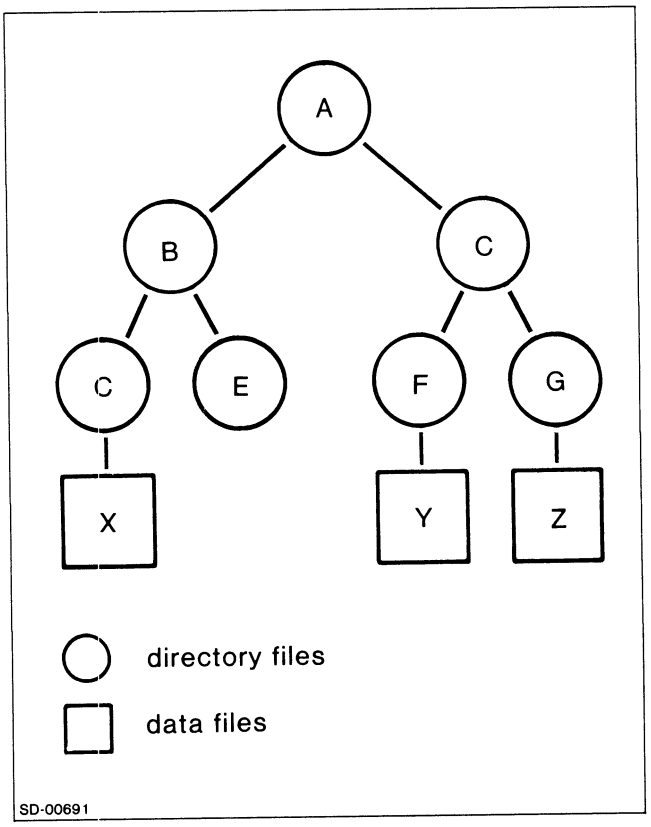


Figure 2-5. A Directory Subtree

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.

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 matched by a filename template. The CLI will access each file that matches the filename template preceding the backslash, except those files that match the filename template *following* the backslash. For example, the template

FILE+\FILE1+

matches every filename in the working directory which 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 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:

=#\T
=+\T:#\T
=+\T:+\T:#\T

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

Colons separate each template in a pathname, and the backslash modifies only the template immediately 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).

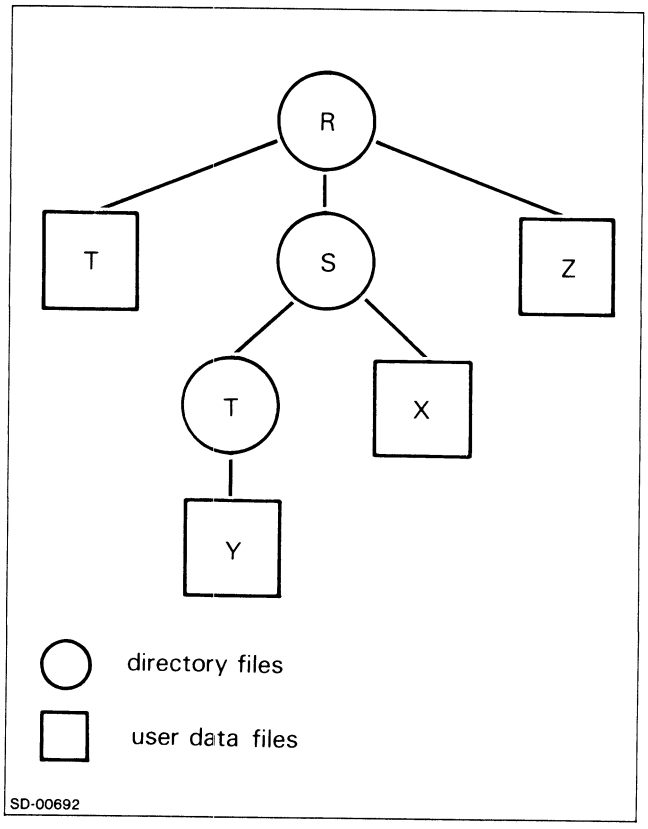


Figure 2-6. Another Directory Subtree

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 in the last example that 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 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.

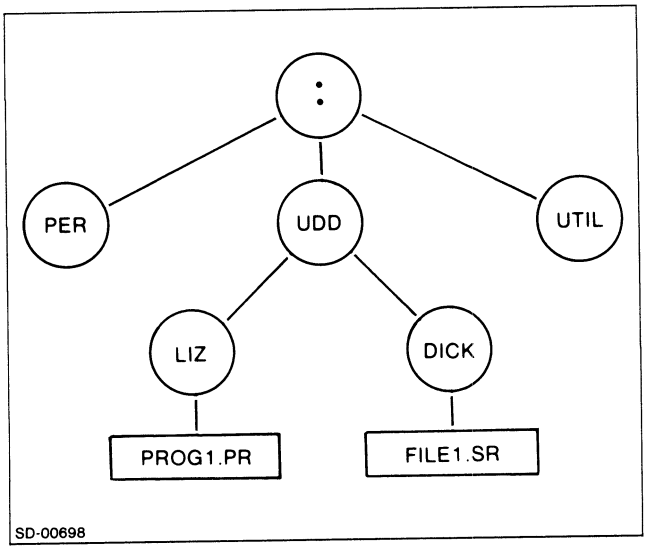


Figure 2-7. Search List Example

To keep most file referencing 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). Suppose your search list is :UTIL,:PER,: and your working directory is DICK. When you type

```
) XEQ MASM FILE1↓
```

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

You can prevent the system from searching directories in your search list by using pathname prefixes. The system will not scan the search list if the pathname has a prefix. Thus, by using the = prefix, you can restrict the filename search to your working directory. You might want to do this, for example, if you want to check whether a file has been created in the working directory.

Links

Another facility that simplifies file referencing is the link. A *link* is a file that contains a pathname or a pathname segment. When a linkname appears in a pathname, the system replaces the linkname with the contents of the link file. (Exceptions to this replacement rule are described below.) You create links with the CREATE/LINK command, and two arguments: the linkname and the pathname that you want to place in the file.

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.

When the pathname in the link file begins with a pathname prefix, a special kind of link resolution occurs. 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'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 do 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 system to start the resolution pathname over again, beginning this time 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 up-arrow (↑) pathname prefix. If a link file contains a pathname beginning with an up-arrow, it is resolved relative to the working directory, 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 up-arrow (↑). 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 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. A few commands, however, operate on the link file itself, not on the file specified by the resolution pathname. Such commands include DELETE, DUMP, FILESTATUS, LOAD, and MOVE. For example, if DELETE has a link filename as an argument, the link file itself is deleted. Similarly, FILESTATUS displays the status of the link file, and MOVE moves the link file.

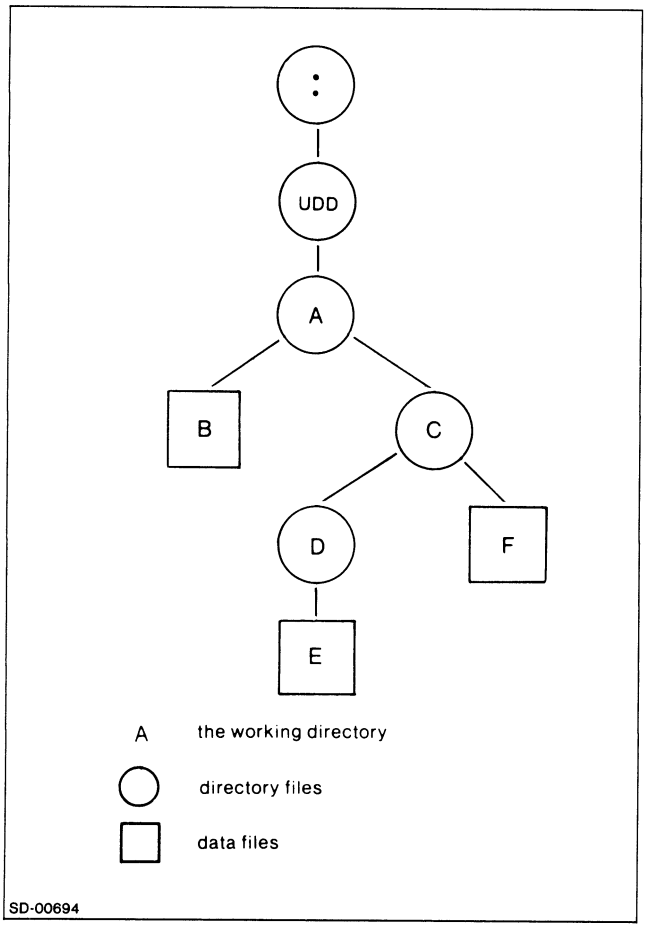


Figure 2-8. Link Examples

Access Control List

Both AOS and AOS/VS allow you to restrict access to any file by stipulating conditions for the file's use. 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 the types of access each user is entitled to. Five types of file access follow:

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

The effect of each access type, as shown in Table 2-2, depends on whether the file is a directory or nondirectory file.

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...]

Each user has the types of access that follow the *first* template matching his username. If a username does not match any template in a file's ACL, then the user has no access to that file. For example, assume that the ACL for FILE1 is as follows:

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

Table 2-2. Access Types

Access	Abbreviation	Non-Directory File	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 file	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

The user JOHNADAMS has Execute access, and the user ADAMSMITH has Read, Write, and Execute access to this file. Note that even though the third entry in the ACL gives JOHNSMITH all access types, he is allowed only Execute access 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. The ACL for FILE1 would be better (from JOHNSMITH's point of view) if it was

```
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 a system utility creates a file for you, its ACL is your default ACL. You can use the ACL command to change and display the ACL of all files to which you have Owner access.

Generic Filenames

Generic filenames simplify the use of certain common files and devices. By using a generic filename, a process need not name specific files. For example, a process may simply open @LIST, and the system will open the process's current LISTFILE (see Chapter 4 for a discussion about the LISTFILE). The LISTFILE could be the line printer, a disk file, or some other output device.

The following are generic files.

- @CONSOLE
- @INPUT
- @OUTPUT
- @LIST
- @DATA

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, @LIST is a disk file created by EXEC named username.LIST.sequence-number; @INPUT is a disk file; and @OUTPUT is the file you specify with the switch /QOUTPUT= when you queue a job. If you omit the /QOUTPUT= switch, @OUTPUT is the line printer.

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

Labeled Magnetic Tapes

A labeled magnetic tape contains your data plus information about your data. Labels of a tape include information such as volume name, filename, file format of the files on the tape (see Figure 2-9). Labeled tapes provide two significant benefits:

- They retain your file information in a consistent format
- You can call your files by a logical name rather than a device name

In order to use labeled tapes you must be logged on under the EXEC either in BATCH or at a user console. Note that this means you may not use labeled tapes from the operator's console.

Labels come in two types: system labels and user labels. The AOS and AOS/VS systems will automatically generate system labels from the information you supply when you create your file. If you want to store additional file information, you may also specify the contents of your own user labels.

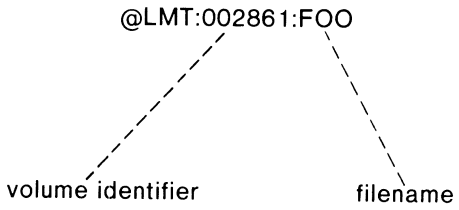
You can use two types of user labels: header and trailer. Both types are useful when you want to specify some information about a file that doesn't fit into the usual label categories; e.g., file creation time, machine configuration. At the highest levels of label processing, you can specify up to nine labels for each type of user label.

To create a labeled tape, you will have to supply the Label utility (Chapter 6) with the following information:

device-name volume-identifier

Once you've created the label, you can reference the tape by using the labeled tape file type (see Table 2-3).

You reference files on labeled tape by appending the volume identifier and filename. For example:



For more information on creating and referencing labeled tapes, see the Label utility in Chapter 6.

For a complete discussion of labeled tapes, see the AOS or AOS/VS programmer's manual.

Device Names and Queue Names

Directory PER lists all system device names and queue names. When you refer to them, you must always use the pathname prefix @ at the beginning of the following pathnames:

@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
@LPT, @LPT1	The first and second line printer (queue names)
@LMT	Labeled mag tape unit
@MTA, @MTA1	(AOS only) Magnetic tape controller 0 or 1 (device names)
@MTB, @MTB1	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

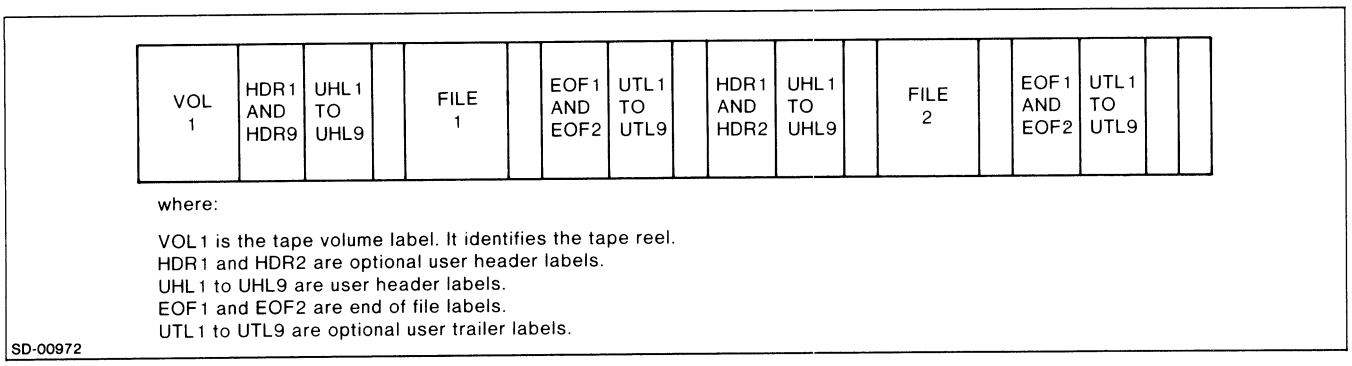


Figure 2-9. Labeled Magnetic Tape

Since a reel of magnetic tape may contain two or more files, you must append the file number to the pathname for these devices. For example, @MTA0:0 names the first file on magnetic tape unit 0, and @MTA0:1 names the second file on magnetic tape unit 0.

Note that you may create a file with the name LPT. The system will not confuse your file named LPT with the line printer because LPT (your file), will not be in :PER.

File Types

The operating system provides 256 types of files. Table 2-3 lists the file types 0 through 127 that are defined by AOS and AOS/VS. 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).

Table 2-3. File Types

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

Table 3-1. Command Switches

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

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 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 command execution. 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.

Certain commands require switch information pertaining to date and/or time. In such cases, you must input the date in the following format:

dd-mmm-yy

For example:

12-DEC-80

Notice you must use a 3-letter abbreviation (e.g., DEC) to identify the month, not a numeric value.

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

The two operators the CLI provides to help you code command lines are parentheses and angle brackets. Parentheses cause command repetition and angle brackets cause argument expansion.

Parentheses

If you enter a command followed by an argument list enclosed in parentheses, the CLI executes the command on all arguments in the list as if each argument were entered on a separate command line. 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 remaining arguments. For example:

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

is equivalent to

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

If you enter two or more commands enclosed within parentheses and followed 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 argument groups 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 a group is exhausted, the CLI executes the command on the remaining arguments in 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 isolates the nested set from the outermost set. That is, the CLI treats the nested 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
```

If you enter a command line containing parentheses nested to three or more levels, the CLI will interpret the outermost parentheses as a repetition operator, the second-level parentheses as a grouping operator, the third-level as a repetition operator, the fourth-level as a grouping operator, etc. You can nest parentheses to any depth.

Angle Brackets

Angle brackets can help you code arguments that contain the same character or character combination. 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 group enclosed by brackets 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 second group is exhausted, the CLI takes the second element from the first group, adds the adjacent character(s), and combines it with the first element in the second group. This second iteration continues until the last argument in the first group is combined with the last argument in the second group, to produce C_Y. The total number of arguments formed 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 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 first expands the argument list by processing angle brackets from left to right. Next, the CLI 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 from left to right, in pairs. 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.

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-80
)
```

The CLI processes a multiple command line from left to right after the entire line is delimited. 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 to the right of the offending command. For more on error handling, see Chapter 4.

Continuation Lines

You can continue a command line to another input line by typing an ampersand (&) before 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; therefore, you must precede the ampersand or begin the continuation line with a separator if required.

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. Once you have continued 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.

EXECUTE or XEQ Command

All utilities are invoked with the EXECUTE or XEQ command. (EXECUTE and XEQ represent the same CLI command.) Some of these utilities can also be invoked by a macro, because the macro contains an XEQ command. Commands and pseudo-macros do not require XEQ.

End of Chapter

Chapter 4
CLI Environments and Exceptional Condition Handling

Introduction

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 the default settings for your CLI process. You can easily check the default settings if you are in the CLI by typing the command CURRENT, which displays each environment parameter setting. In this chapter, we describe each of these parameters and explain 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 increments the LEVEL setting and copies all other parameters to the next level. POP decrements 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, obviously, you have access to the current level's parameters only.

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 the users are entitled to. 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 the current SUPERUSER mode 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 which compose a process's state.)

You can set and display the current SUPERPROCESS mode 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 mode (see the WHO and RUNTIME commands in 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 Chapters 1 and 6.) However, 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 mode is ON, it compresses two or more spaces or tabs into a single blank. When SQUEEZE mode is OFF, as it normally is, spaces and tabs are not compressed.

You set and display the squeeze mode with the CLI command SQUEEZE (see Chapter 6). You use the /Q switch to override the current squeeze setting for the duration of any CLI command.

CLASS1

CLASS1 is described in the next section under exceptional conditions.

CLASS2

CLASS2 is described in the next section under exceptional conditions.

TRACE

When debugging a large macro, it is useful to enable tracing. With tracing on, you can see the full expansion of all macros, pseudo-macros and commands as they are being processed. You set and display the TRACE mode with the CLI TRACE command (see Chapter 6).

Variables

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

You set and display the current value of the variables with the ten CLI commands VAR0 through VAR9. You can insert the value of the variables into a command line with the ten 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. Then, before you execute the program, you can set LISTFILE to any file you wish. For example, you can have your program write to @LPT for one run and @MTA0:0 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 listfile 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, you 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 card reader on one run and from a magnetic tape unit on another run.

You set and display the current datafile 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).

LOGFILE

LOGFILE contains the pathname of the current logfile. When logging is on, all input to @INPUT and all output to @OUTPUT is copied into the logfile. (Output from the TYPE command, however, is not 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 logfile with the LOGFILE command.

Working Directory

The working directory is your reference point 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 which 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 the 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. You can divert a program's termination message (which is 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.

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 holds a maximum of eight CLI commands. By default, the prompt is null.

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 characteristics of your console are preserved in your environment.

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 the CLI environment if it 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.

Using CLI Environment Levels

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.

```
PUSH
SQUEEZE OFF
CLASS1 ABORT
CLASS2 WARNING
LISTFILE %1%
DATAFILE %2%
DIRECTORY :UDD:USER:MDIR
SEARCHLIST :UDD:USER:NDIR [!SEARCHLIST]
XEQ/S MYPROG1
[!EQUAL,[!STRING],2]
XEQ MYPROG2
[!ELSE]
XEQ MYPROG3
[!END]
POP
```

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 is a dummy argument. (Chapter 5 explains dummy arguments.) The CLI replaces %1% by the first argument and %2% by the second argument you supply when you call the macro. For example, you might call this macro with the CLI command line:

```
) SAMPLE @LPT @MTA0:1↵
```

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 when the macro is called.

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 POPs to the previous environment, and then exits from the macro. Before returning to interactive mode, the CLI restores the previous level's settings.

End of Chapter

Chapter 5
CLI Macros

Introduction to CLI Macros

A macro (short for macroinstruction) is a command that stands for a sequence of commands. Normally, you store a CLI macro in a file and execute it by typing the filename. Suppose, for example, that you have a file called TDP.CLI which 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-80
YOUR PROCESS NUMBER IS 15
)
```

If there is a sequence of CLI commands that you execute frequently, you can save time by putting them 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 code paths in the same macro depending on the values of variables.

Creating Macros

To create a macro, just enter commands into a file. The macroname is the filename (minus the .CLI extension if you use it). There are two ways to enter commands into a file. The first is to use a text editor; e.g., SED, LINEDIT or SPEED. If you use a text editor, simply enter the commands into the file in the order you want them executed. Be sure to end all command lines with a valid delimiter.

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 get out of input mode, type another right parenthesis next to the double prompt and then hit the NEW LINE key.

Example

To create the TDP.CLI macro listed above, 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).

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 first for a filename with the macro name plus the .CLI extension. 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 look for a file whose name is TDP.CLI. If it doesn't find a file with that name, it will look for a filename without the .CLI extension (i.e., TDP). If you type:

```
) TDP.CLI↵
```

the macro will still be executed because the CLI, after not finding a file with filename TDP.CLI.CLI, would then look for, and find, the file TDP.CLI.

Be careful that your macro names are not the same as CLI commands. If there is an ambiguity, the CLI will execute the CLI command and not the macro, unless you enclose the macro name in square brackets.

Dummy Arguments

Dummy arguments are variables used by macros to represent actual arguments. By *actual* argument, we mean the string or number which a command 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 CLI will not actually try to delete a file called %1%. Instead, it will decide what file %1% represents and delete that file. Dummy arguments are always enclosed by percent signs which is how the CLI distinguishes them from actual arguments. Moreover, dummy arguments are only used in macros. A dummy argument can represent one actual argument, a range of actual arguments, or a switch. Table 5-2 summarizes the format of the three types of arguments.

Single Dummy Arguments

The simplest type of dummy argument is one containing a single number enclosed by percent signs. The number represents the numerical position of the actual argument with which the macro was called. 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:

```
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 following commands will be executed:

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

The net result will be that the two files -- FILEA and FILEB -- will have their names exchanged. If you had called the macro with only one argument (i.e., SWITCH FILEA), then %2% would have been 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 *m* is omitted, 1 is assumed; if *n* is omitted, 32767 is assumed; if *i* is omitted, 1 is assumed (see Table 5-1). Hence, %1-4,1%, %1-4%, and %-4% are all equivalent. If the macro was only called with 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 uses slashes and backslashes to signify the inclusion or exclusion of switches in a dummy argument. In general, 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. The examples below illustrate how this works. If a switch takes a value (e.g., /COPIES=5), its value is automatically included in the expansion.

Switch dummy argument	What it represents
%1/S%	First argument's S switch
%2\L%	All of the second argument's switches except L
%0/%	All of the macro-name's switches
%1/W/ff/G%	The W, ff, and G switch of the first argument. (The CLI is not case sensitive, so it would match these switches even if they had been called as /w/FF/g.)

There is also a dummy format to extract only a switch value rather than the entire switch. The general 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 MAIL 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 5-2. 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)
%m-n,i%	Expands to every ith argument from the mth to the nth
%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. If the /s switch takes a value, the value is included in the expansion
%n/s=%	Expands to the value of the nth argument's /s= switch
%n\s%	Expands to all of the nth argument's switches except /s

Macro Syntax

In general, the syntax for macros is the same as the syntax for CLI commands described in 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 macro ASM.CLI contains the following command lines:

```
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.

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 which 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 each word in the expression *INPUT FILE ERROR* is treated 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 since it may expand to more than one word.

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. 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 you have a file called CONTENTS which contains the following line:

```
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 lineprinter.

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 code 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. 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 evaluate to 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 evaluate to unsigned decimal integers.

Each conditional section of a macro must begin with one of these pseudo-macros. It must end with an [!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 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]. Figure 5-1 illustrates the routing for conditional pseudo-macros.

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 which expands to either ON or OFF (see Chapter 6). The resulting expansion will be compared 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. The two WRITE command lines are included 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
)
```

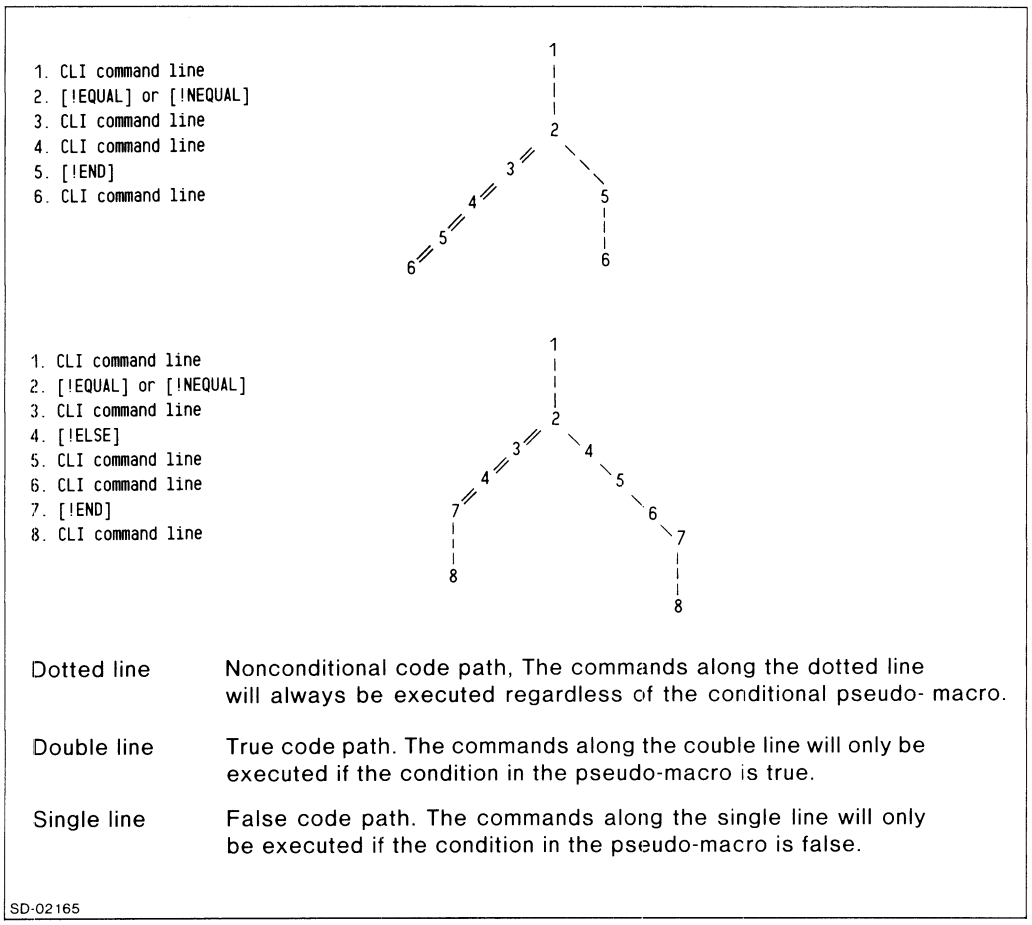


Figure 5-1. Conditional Code Paths

Conditional Arguments

The two arguments that you pass to [!EQUAL] and [!NEQUAL] are treated as strings and compared character by character. If one string is longer than another, they are considered 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.

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 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 5-3).

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

Table 5-3. 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 to 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.
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 all conditional pseudo-macros have been accounted for. 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 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 which do not fall within a conditional routine, the CLI will issue an error message.

Figure 5-2 illustrates how the CLI evaluates nested pseudo-macros.

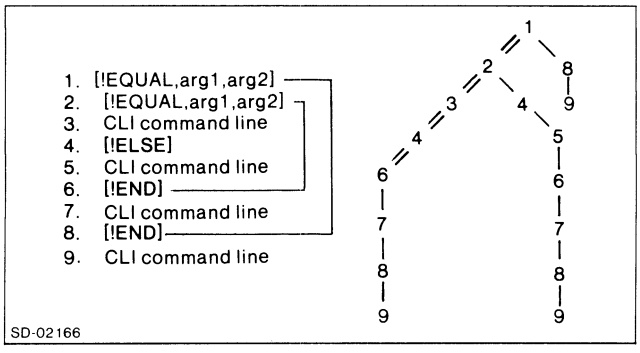


Figure 5-2. Nested Conditionals

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 and they will be executed 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 executes the macro and returns a normal prompt.

The MAIL Macro

The MAIL macro (see Figure 5-3) allows users to send mail to one another through the computer system. You can send mail to another user whether he/she is logged on or not. However, the recipient of the mail will only be notified that there is mail when he/she executes this macro. Note that the macro assumes the existence of a :MAIL directory, which is shared by all users.

The actual executable macro does not have the numbers in front of each command line -- we have put them there only as a reference aid. We have indented the lines to improve readability. The indentations are not necessary, but the CLI ignores any spaces or tabs at the beginning of command lines. Because of the limited width of the page, we have had to carry several command lines onto a second line, using the continuation character (&).

The MAIL macro will perform different operations depending on which switch you call it with. Table 5-4 shows the format for the various mail macro command switches. Arguments enclosed by brackets are optional.

Table 5-4. MAIL Macro Syntax

Macro Call	Result
MAIL *[username(s)]*	MAIL without a switch will list any mail sent to you. If you include arguments, then MAIL will list all mail sent to you by the user(s) that you specify.
MAIL/S username *[filename]*	Send mail to a user. If you do not specify a filename with the MAIL/S macro call, the CLI will issue a double prompt to inform you that it is waiting for input into the mail file. After you have finished writing your message, type a right parenthesis followed by NEW LINE. You can send the same mail to more than one user by enclosing the list of usernames in parentheses and specifying a filename. If you include the filename argument, the contents of the file will be sent as the mail.
MAIL/R *[username(s)]*	Read mail sent to you. If you specify a username, then the CLI will display mail from that user onto your terminal. If you don't specify a username, then MAIL/R will display all outstanding mail.
MAIL/L*[=filename]*	Send mail to the current listfile (/L) or to a file or device designated by /L=filename. If you use this switch, the mail will not be displayed on your terminal. You must also include /R for the /L switch to be enabled.
MAIL/D *[username(s)]*	Delete any mail sent to you by username(s). If you don't specify username(s), then MAIL/D will delete all mail sent to you.
MAIL/F *[username(s)]*	Delete all mail that you have sent to username(s). If you don't specify username(s), then MAIL/F will delete all mail that you sent.

Examples

Suppose user Colleen writes:

```
) MAIL/S JOHN↓
)) Can we meet to discuss the project?↓
)) Let me know if you're free at 3:00.↓
))↓
)
```

This will send the mail to user John. Now if John types:

```
) MAIL↓
```

The CLI will return:

```
DIRECTORY      :MAIL

 JOHN_FROM_COLLEEN 6-JUL-80 11:17:26 261

*** USE "MAIL/R" TO RECEIVE MAIL ***

)
```

To read the mail, John types:

```
) MAIL/R↓

**************************************
Sent by COLLEEN at 9:31:04 on 02-JUN-80
**************************************
Can we meet to discuss the project?
Let me know if you're free at 3:00.
************ END OF LETTER ************

END OF MAIL

)
```

To delete the mail, John types:

```
) MAIL/D↓
DELETED :MAIL:JOHN_FROM_COLLEEN
)
```

```
1   PUSH
2   [!NEQUAL.%0/F%.]
3     WRITE
4     DELETE/V/2=IGNORE &
          :MAIL:[!EQUAL.%1%.]+[!ELSE]<%1-%>[!END]_FROM_[!USERNAME]
5     WRITE
6     [!ELSE]
7     [!EQUAL.%0/S%./S]
8       [!EQUAL.%1%.]
9         WRITE
10        WRITE *** NO ADDRESSEE SPECIFIED ***
11        WRITE
12        [!ELSE]
13        CLASS2 IGNORE
14        STRING :MAIL:%1%_FROM_[!USERNAME]
15        CREATE [!STRING]
16        PUSH
17        [!EQUAL.%2%.]
18          STRING :MAIL:[!PID].MAIL.TMP
19          DELETE [!STRING]
20          CREATE/I [!STRING]
21          [!ELSE]
22          STRING %2%
23          [!END]
24        WRITE/L=[!STRING/P] ****************************************
25        WRITE/L=[!STRING/P] Sent by [!USERNAME] at [!TIME] on [!DATE]
26        WRITE/L=[!STRING/P] ****************************************
27        COPY/A [!STRING/P] [!STRING]
28        WRITE/L=[!STRING/P] ************* END OF LETTER *************
29        [!EQUAL.%2%.]
30          DELETE [!STRING]
31          [!END]
32        POP
33        ACL [!STRING] %1% OWR [!USERNAME] OWR
34        [!END]
35      [!ELSE]
36      [!EQUAL.[!LOGON].CONSOLE]
37        STRING ([!FILENAMES &
            :MAIL:[!USERNAME]_FROM_[!EQUAL.%1%.]+[!ELSE]<%1-%>[!END]])
38        [!EQUAL.[!STRING].]
39          WRITE No Mail.
40          [!ELSE]
41          [!EQUAL.%0/%.]
42            FILESTATUS/SORT/TLM/LENGTH &
                :MAIL:[!USERNAME]_FROM_[!EQUAL.%1%.]+[!ELSE]<%1-%>[!END]
43            WRITE
44            WRITE *** USE "MAIL/R" TO RECEIVE MAIL ***
45            WRITE
46            [!ELSE]
47              [!NEQUAL.%0/R%.]
48                CHARACTERISTICS/PM
49                TYPE%0/L% &
                    :MAIL:[!USERNAME]_FROM_[!EQUAL.%1%.]+[!ELSE]<%1-%>[!END]
50                WRITE%0/L% END OF MAIL
51                CHARACTERISTICS/OFF/PM
52                [!END]
53              [!NEQUAL.%0/D%.]
54                DELETE/V/2=IGNORE  &
                    :MAIL:[!USERNAME]_FROM_[!EQUAL.%1%.]+[!ELSE]<%1-%>[!END]
55                [!END]
56              [!END]
57            [!END]
58          [!END]
59        [!END]
60      [!END]
61    POP
```

Figure 5-3. The MAIL Macro

1. Push to the next level. This preserves the current environment's settings.

2. If the /F switch to the macro does not equal a null (i.e., if there is an /F switch), then execute the following commands up to an [!ELSE] or [!END]. If there is no /F switch, then skip to the first [!ELSE] or [!END] that does not match up with another [!EQUAL] or [!NEQUAL].

3. WRITE without an argument will output a NEW LINE.

4. The DELETE command has a conditional argument. The /V switch tells the CLI to display the name of each deleted file on the terminal. The /2=IGNORE switch tells the CLI to ignore any CLASS2 errors (this is to prevent the CLI from displaying an error message if it tries to delete a file that does not exist). The logic of the conditional argument runs as follows: if there is no first argument, then the CLI will delete +_FROM_[!USERNAME], i.e., all files in the directory which end with _FROM_(the user's username). If there is a first argument, then the CLI will delete any file in the directory which matches (any of the arguments)_FROM_(the user's username). For example, if the user was PAUL and the only argument to the MAIL macro was STEVE, then this command would delete the file STEVE_FROM_PAUL if it existed.

5. Write a NEW LINE.

6. This [!ELSE] goes with the [!NEQUAL] on line 2. If you called the macro without a /F switch, then execution would begin here. If there was a /F switch present, then execution flow would skip to line 60.

7. If the macro was called with a /S switch, then execute the following commands. Otherwise, skip to line 35. Note that if the /S switch had been given a value (/S=5 for instance), then the program flow would skip to the [!ELSE] on line 35.

8. If there are no arguments, then execute the following commands. If there is at least one argument, then skip to line 12.

9. to 11. Write the message to the terminal and then skip to line 34.

12. If there is an argument (see line 8), then execute the following commands.

13. Set CLASS2 errors to IGNORE status (see line 4 for the purpose of this command).

14. Set STRING to :MAIL:(first argument)_FROM_(user's username).

15. Create a file under the MAIL directory whose filename is given by the current value of STRING, that is, (first argument)_FROM_(username).

16. Push to the next level. This saves the current value of STRING.

17. If there is no second argument, then execute the following commands, which create a temporary file for the mail to be sent. If there is a second argument (which specifies the file to be sent), then skip to line 21.

18. Set STRING to :MAIL:(user's ID number).MAIL.TMP.

19. Delete any file whose filename matches STRING (see previous command). Note that no error will be returned if no such file exists because the error status was set to IGNORE in line 13.

20. In input mode, create a file with the same filename as the one you just deleted.

21. If there is a second argument (see line 17), then execute the next command.

22. Set STRING to the second argument.

23. This [!END] corresponds to the [!EQUAL] on line 17.

24. to 26. Write out a message to the file whose filename matches the previous environment's STRING. The last level change occurred on line 16, so that the previous STRING is the one set on line 14. Note that the file must exist, since we created it on line 15.

27. Append the contents of the file :MAIL:(first argument)_FROM_(user's username) to the file denoted by the current environment's STRING (as it was set in either line 18 or line 22, depending on whether there was a second argument.)

28. Same as lines 24-26.

29. If there is no second argument, then execute the following commands. Otherwise, skip to line 31.

30. Delete the file represented by STRING.

31. This [!END] corresponds with the [!EQUAL] on line 29.

32. Pop to the previous level.

33. Set the ACL of the file represented by STRING (set in line 14) to (first argument) OWR (username) OWR. The only people who will have owner, write and read access are: the person to whom the message was sent (first argument), and the person who sent the message (username). Note that if there is no second argument, then the file represented by STRING would have been deleted in line 30, and we would be trying to change the ACL of a non-existent file. This will not generate an error message, though, because we have set CLASS2 to IGNORE in line 13.

34. This [!END] corresponds to the [!EQUAL] on line 8.

35. This [!ELSE] corresponds to the [!EQUAL] on line 7.

36. The [!LOGON] pseudo-macro returns one of three values. If the process is not logged-on under EXEC, nothing is returned. If the process is logged-on under EXEC, then the pseudo-macro returns either BATCH (if the process is logged-on in a batch stream), or CONSOLE (if the process is logged-on at a console). [!EQUAL,[!LOGON],CONSOLE] tells the CLI to execute the subsequent commands if the process is logged-on at a console. If not, execution flow will skip to line 58. This command avoids executing the rest of the macro if the process is logged-on under batch. When you queue up a job, you won't want the CLI to check for mail.

37. This command line involves a conditional argument. Up to the conditional pseudo-macros, the command looks like this:

    STRING ([!FILENAMES &
    :MAIL:[!USERNAME]_FROM_

    This will set STRING equal to a filename beginning with [!USERNAME]_FROM_. What comes after the last underline depends on the first argument with which the MAIL macro was called. If there was no first argument, then the [!EQUAL] pseudo-macro is true and the plus template will be appended to the pathname. The expression will then expand to all filenames beginning with [!USERNAME]_FROM_. If, on the other hand, the MAIL macro was called with one or more arguments, then the [!ELSE] will be executed. The expression following the [!ELSE] is <%1-%>, which expands to all arguments. The angle brackets cause the CLI to generate a complete pathname for each argument instead of simply appending them all to the end of [!USERNAME]_FROM_. Note also the parentheses surrounding the entire string expression. They direct the CLI to execute the command line once for each argument--that is, for each filename that the expanded argument matches. STRING will therefore be equal to the last argument matched, or it will be null, if no filenames match the expanded argument. STRING need contain only one filename, because its purpose is to state whether there is at least one desired file in :MAIL (see line 38).

38. to 39. If STRING is null, execute the next command, which write "No mail." to the terminal.

40. If there were files that matched the pathname in line 37, then execute the following commands.

41. If there were no switches on the MAIL call, then execute lines 42 through 45. If there were switches, then skip to line 46.

42. Display the name and filestatus of all files that match the conditional argument (the comment on line 37 explain the conditional argument). The switches on the FILESTATUS command tell the CLI to sort the files alphabetically, give the time the file was last modified, and its length.

43. to 45. Write out message to terminal.

46. If the mail macro was called with a switch, then continue execution here.

47. If there is a /R switch, then execute the next command. Otherwise, skip to line 52.

48. Set the page mode characteristic on. This will prevent the message from rolling off the screen if it is more than one screen long (if you have a video display terminal).

49. to 50. These commands actually print the mail. The files it types depend on the first argument with which the mail macro was called (see line 37).

    If you included a /L switch in the macro call, the dummy switch %0/L% expands to /L or /L=pathname on the TYPE and WRITE commands. As result, the output is directed to the specified pathname or to the generic @LIST file. If there was no /L switch in the macro call, the dummy switch expands to null, and has no effect on the command. The file will be typed to the generic @OUTPUT file, which is usually the terminal.

51. Turn page mode off.

52. This [!END] corresponds to the [!NEQUAL] on line 47.

53. If the /D switch does not equal null (i.e., if there is a /D switch), then proceed to the next command. Otherwise skip to line 55.

54. Delete the file whose filename matches the conditional argument (The comment on line 37 explains the argument). The /V switch tells the CLI to print out the name of each deleted file.

55. This [!END] corresponds to the [!NEQUAL] on line 53.

56. This [!END] corresponds to the [!EQUAL] on line 41.

57. This [!END] corresponds to the [!EQUAL] on line 38.

58. This [!END] corresponds to the [!EQUAL] on line 36.

59. This [!END] corresponds to the [!EQUAL] on line 7.

60. This [!END] corresponds to the [!NEQUAL] on line 2.

61. Pop to the previous level. This makes up for the PUSH on line 1, and restores the environment that existed when the macro was called.

End of Chapter

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

This chapter presents descriptions of CLI commands, pseudo-macros, 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 section, we use both conventional brackets [ ] and italic brackets and entries *[entry]*. The italic brackets and entries indicate optional entries; the conventional brackets do not. Conventional brackets are part of the command; you must enter them.

System Utilities

This chapter also reviews the most frequently used system utilities. It contains brief descriptions of the CLI command lines used to invoke these utilities. Many of these utilities have their own manuals which contain more complete descriptions. To invoke a utility, begin the command line with the XEQ (or EXECUTE) command; this creates a process to execute the utility and blocks the calling process. (Note that some of the utilities 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 has finished. To interrupt an executing utility, see "Control Characters" in Chapter 1.

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

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

In all but the utilities 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, and Utility 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.
MOUNT linkname message	Mount a tape.
MOVE dest-dir *[sourcefile]...*	Move copies of one or more files.
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)

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

Process Control

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, and Utility Summary by Category

CLI Environment (continued)

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, and Utility 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.
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
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, and Utility 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, and Utility Summary by Category

Pseudo-Macros

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, and Utility Summary by Category

System Utilities	
Utility	**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.
CLINK	(AOS/VS only.) Bind 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 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	Bind object modules into an executable FORTRAN 77 program.

(continued)

Table 6.1 Command, Pseudo-macro, and Utility Summary by Category

System Utilities	
Utility	**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 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. The arguments must be separated by a 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

/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

/K
Delete ACL; this denies everyone except Superusers access to the file until the ACL changes again (takes pathname argument only)

/D
Give the file the user's default ACL (takes pathname argument only)

Argument Switches

None

Examples

```
) 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 *Utility*

Invoke the APL interpreter (AOS/VS only).

Format

XEQ APL *[initial-workspace-pathname]*

You may specify an initial workspace. If you do not, the interpreter looks by default for a workspace named CONTINUE. If this file exists, it is automatically loaded. 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 **term-type** value:

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

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. It then 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.

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. The possible values for **term-type** are listed above
/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. The possible values for **term-type** are listed above
/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. The possible values for **term-type** are listed above
/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 to use

APL (continued)

Example

```
) XEQ APL/L=LOGGING.FILE PROG2↓
```

Invoke the APL interpreter, loading PROG2.WS as the initial workspace. In addition, use LOGGING.FILE as the log file.

!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 wouldn't normally be interpreted correctly by the CLI. 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).

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

```
) WRITE [!ASCII 207]↓
)
```

ASSIGN *Command*

Assign a character device for your exclusive use.

Format

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

character-devices include the card reader, printer, 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 @CRA↵
)
      .
      .  (you have exclusive use of the device)
      .
) DEASSIGN @CRA↵
)
```

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. Each is designed to run under different operating systems.

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↵
)
```

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 zero and the default maximum is 'no limit'. If you set only the minimum number, the maximum will automatically be set to 'no limit'. See the *AOS System Manager's Guide* (093-000193) or *Managing 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

Examples

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

Display the system's bias factor.

```
) BIAS 0↓
)
```

Set 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.

objectmodule is the name of the first object to be bound. Unless you specify otherwise with the /P switch, the program file will be named 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. These will be placed 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 which 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 though a listing is specified
/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 the name of the first object module
/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 be bound 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, records extracted from the library will be bound 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
symbol/V=integer	Assign value integer to this accumulating symbol, which was defined by pseudo-op .ASYM
/X	(Used in conjunction with the /B switch.) Exclude this shared library routine, which is in the library included by the /B switch
integer/Z	Set the ZREL base to this octal integer. If the current ZREL base exceeds this value, the system ignores the switch

BIND (continued)

Examples

```
) 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. Because the overlays are enclosed in one set of brackets, the system will reserve one overlay area in memory for them. 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

Examples

```
) BLOCK 19↵
)
```

Block the process with PID (process ID) 19.

```
) BLOCK SMITH:PROG1↵
)
```

Block 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 pc) 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 pc symbolically.

BYE *Command*

Terminate this CLI process.

Format

BYE *[argument]...*

The CLI returns any *arguments* 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

BYE (continued)

Argument Switches

None

Example

```
) BYE↵
AOS CLI TERMINATING 26-NOV-80 13:00:19
```

Terminate the CLI. If the CLI's father was EXEC, you are logged off the system. (Note that, under AOS/VS, the termination message reads *AOS/VS CLI TERMINATING.*)

CBIND *Utility*

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

Format

CBIND objectmodule *[argument]...*

CBIND is a macro that 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 though another listing file is specified
/H	List all numbers in hexadecimal
/I	Build a nonexecutable program file, lacking a UST, TCBs, and all other system databases
/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
/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, which is required 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↓
```

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

CHAIN *Command*

Overwrite your CLI with the program named in pathname and transfer CPU control to that program's entry point.

Format

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

The arguments are placed in the initial inter-process communication (IPC) message to the new process. The new process can access these arguments through the ?GTMES system call. See Appendix B for details. If you are on an AOS/VS system, note that a 32-bit program cannot chain to a 16-bit program, and a 16-bit program cannot chain to a 32-bit program.

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
/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
/D		Enter the Debugger

Argument Switches

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

Examples

) CHAIN MYPROG↵

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

) CHAIN MASM/L TESTA↵

Load MASM into memory and begin execution at its entry point. /L produces a listing file.

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.* 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

/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

/LPP=n — Lines per page, in decimal

/CPL=n — Characters per line, in decimal

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

/RESET — Set the characteristics of a device to its default characteristics. This switch must be used alone

/PREVIOUS — Set the current characteristics of a device to the previous environment's characteristics (no arguments or other switches allowed)

/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

/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

/8BT — All 8 bits of an ASCII character are interpreted 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

/EB0 — For echoing to occur on your terminal, you must set /EB0 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)

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

CHARACTERISTICS (continued)

/EOL	Do not output a new line if CPL line length is exceeded on output
/ESC	ESC character produces ↑C↑A 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)
/LT	(AOS only) Output 60 (decimal) nulls on open and close
/MOD	Device is on a modem interface
/MRI	(AOS/VS only) Monitor Ring Indicator
/NAS	If this switch is ON, set non-ANSI standard bit. The device is considered non-ANSI standard. 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
/OTT	On input, convert octal 175 and 176 to octal 33
/PBN	Packed format on binary read, 4 columns are put in 3 words; if this switch is OFF, columns are right-justified in memory (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
/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)
/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
/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
/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

Examples

```
) CHARACTERISTICS↵
/HARDCOPY/LPP=24/CPL=80
/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 has terminated abnormally (e.g., console interrupt, trap, etc.), 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
/Q	Set SQUEEZE to ON for this command
/P	Set CLASS1 severity level to the previous environment's severity level (no arguments allowed)

Argument Switches

None

Example

```
) CLASS1↵
ERROR
) CLASS1 ABORT↵
)
```

First, display the current CLASS1 setting, then change 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
/Q	Set SQUEEZE to ON for this command
/P	Set CLASS2 severity level to the previous environment's CLASS2 severity level (no arguments allowed)

Argument Switches

None

Example

```
) CLASS2↵
WARNING
) CLASS2 IGNORE↵
)
```

First, display the current CLASS2 setting, then change it to IGNORE.

CLINK *Utility*

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 *Utility*

Compile a COBOL source file.

Format

For AOS:

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

For AOS/VS:

COBOL source-pathname

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 to be assigned to the object file the compiler produces. 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).

Two sets of COBOL switches are given below, 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)
/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 no pathname is specified, write error messages to @OUTPUT. If a pathname is specified, write error messages to that file
/ERRORCOUNT=integer	Terminate compilation after the specified number of errors. The default value is 100
/HALT	Suppress generation of object code
/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
/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
/MAPCASE	Translate all identifiers into uppercase before compilation
/N	Suppress production of the object file
/NOCOPIES	Suppress printing of all copy files
/NOMAP	Suppress printing of the storage map
/NOWARNINGS	Suppress severity 1 error messages
/NOX	Perform extended arithmetic operations in non-extended mode

COBOL (continued)

/O=pathname	Write the object file to pathname.OB
/OCTAL	Produce octal values for alphanumeric literals
/PROCID	Save the procedure names at runtime, and print the procedure name if a fatal error occurs
/STAT	Write compilation statistics to @OUTPUT
/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*. If you do not use this switch, @LIST is assumed
/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 will be sent to FILE1.ER. Symbol and line information are generated for later use by the SWAT debugger (/DEBUG).

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

Argument Switches

None

Examples

```
) 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
)
```

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 or a system spooler. 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
/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
/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)
/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)

Argument Switches

None

Examples

```
) CONTROL @SPOOL RESTART @LPA↵
)
```

Direct the spooler to restart output on the line printer.

```
) CONTROL @EXEC ENABLE @CON1↵
)
```

Direct the EXEC process to enable @CON1.

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
)
```

Produce 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.

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

COPY (continued)

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

/Q — Set SQUEEZE to ON 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

/D — Delete dest-file (it must exist) and recreate dest-file using the same specifications as for the old dest-file

/IDENSITY=mode — Control the magnetic tape density for input files. Use this switch with MTB, model 6026 tape drives only. These are your mode options:

MODE	DENSITY
800	800 BPI
1600	1600 BPI
ADM	Automatic Density Matching

/IMTRSIZE=block-size (bytes) — Control the magnetic tape block size for input files

/ODENSITY=mode — Control the magnetic tape density of output files. Use this switch with MTB, model 6026 tape drives only. 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.

/V — Display the pathname of the sourcefile copied

Argument Switches

None

Examples

```
) COPY OUTPUTFILE FILEA↵
)
```

Copy FILEA to OUTPUTFILE; create OUTPUTFILE using FILEA's specifications.

```
) COPY/A TESTALL TEST1 TEST2 TEST3↵
)
```

Append TEST1, TEST2, and TEST3 to the end of TESTALL.

```
) COPY TESTA @MTA0:0↵
)
```

Copy file on MTA0:0 to TESTA and use default specifications to create TESTA.

```
) COPY/V TEST1 TEST2↵
TEST2
)
```

Copy TEST2 to TEST1, using the /V switch to display the source file's pathname.

CPUID *Command*

Display the CPU identification (AOS/VS only).

Format

CPUID

This command displays your computer's central processing unit (CPU) identification. For a description of this number, consult the *ECLIPSE® MV/8000 Principles of Operation* manual (014-000648).

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
)
```

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
/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
/DATASENSITIVE		Create the file with data sensitive record format
/DIRECTORY		Create a directory. If the /MAXSIZE= switch is also used, 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

CREATE (continued)

/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 128^2 elements; and so on

/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.

/TYPE=type — Create a file of type **type**.

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

Examples

```
) CREATE/I PROG.FR↓
))DIMENSION ARRAY(100)↓
      .
      .
      .
))END↓
))))↓
)
```

Create a FORTRAN IV source file named PROG.FR.

```
) CREATE/LINK LNAME :UDD:USERNAME:FILE1↓
)
```

Create a link file containing a complete pathname to FILE1.

```
) CREATE/DIRECTORY PROJECT1↓
)
```

Create a directory called PROJECT1.

CURRENT *Command*

Display the current CLI environment's settings.

Format

CURRENT

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

```
) 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
                     /ON/ST/EB0/EB1/ULC/WRP
                     /OFF/SFF/EPI/8BT/SPO
/RAF/RAT/RAC/NAS/OTT/EOL/UCO/LT/FF
/PM/NRM/MOD/TO/TSP/PBN/ESC/FKT/NNL

)
```

Display the current CLI environment's settings.

DATAFILE *Command*

Set or display the current DATAFILE pathname.

Format

DATAFILE *[pathname]*

This command sets the DATAFILE to *pathname*. The DATAFILE is passed 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 create a process, you can specify the DATAFILE *pathname* which you want the CLI to pass as the generic @DATA to the created process.

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
/G		Set the filename to @DATA (no arguments allowed)
/K		Set to null string (no arguments allowed)
/P		Set DATAFILE to previous environment's DATAFILE (no arguments allowed)

Argument Switches

None

Example

```
) DATAFILE↵
@DATA
) DATAFILE MYFILE↵
)
```

First, display 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

Examples

```
) 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, evaluate [!DATAFILE] and write the current data file pathname, which is the generic @DATA. Then change environment, and set a new data file for the new environment. Evaluate and write [!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 username OP (process-ID 2), the initial CLI. Use one of the following formats for *date:*

11 26 80

26-NOV-80

In the second format, you can use an abbreviation if it uniquely identifies 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

Examples

```
) DATE 11 26 80↓
) DATE↓
26-NOV-80
)
```

Set and display the date.

```
) DATE 26-N-80↓
) DATE↓
26-NOV-80
)
```

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-80.
)
```

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 @CRA1↵
)
```

Deassign the alternate card tape reader, that you previously assigned.

DEBUG *Command*

Execute the specified program and enter the 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 arguments are placed 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.

See the EXECUTE command for additional information about subordinate processes. For more on DEBUG, see 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).

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
/I	Create input for the program from @INPUT. The last line of input must contain a single)
/M	Create input for the program from the macro body. The last line of input must contain a single)
/S	Return the termination message to STRING

Argument Switches

These are explained in the *AOS Debugger and Disk File Editor User's Manual* (093-000195) and in the *AOS/VS Debugger and File Editor User's Manual* (093-000246).

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, that 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).

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 DIR1:MYFILE.PR↵
+ START: 001762 (CR or LF)
    .
    .
+ 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
/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
/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
/P		Set default ACL to the previous environment's default ACL (no arguments allowed)

Argument Switches

None

Examples

```
) DEFACL↵
SMITH, OWARE
) DEFACL SMITH,R↵
) DEFACL↵
SMITH,R
) DEFACL/D↵
) DEFACL↵
SMITH,OWARE
)
```

The first command displays the current default ACL, that 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, evaluate the pseudo-macro [!DEFACL], then write 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 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
/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
/V	Verify each deletion with a list of the files deleted

Argument Switches

None

Examples

```
) 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/8000-16	AOS/VS
AOS	MV/8000-16	AOS/VS
16	MV/8000-16	AOS/VS
X16	MV/8000-16	AOS/VS
VS16	MV/8000-16	AOS/VS
X	MV/8000-32	AOS/VS
X32	MV/8000-32	AOS/VS
32	MV/8000-32	AOS/VS
VS32	MV/8000-32	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 syntax errors are found. 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. The legal values for symbol are given above. 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
/F	Produce an error message if the compilation results in an attempt to generate a floating-point instruction

DGL (continued)

/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 will use software multiply/divide)
/I	Do not list the contents of INCLUDE files on the compilation listing
/INNER	Produce code that is to be linked for an inner (4-6) ring, and that can be called 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. That is, 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; that is, 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. Compiling with /P 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 truncation 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

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. This argument switch can be appended 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, to be 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 causes additional information about the lines of code to be written 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
/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
/I		Set working directory to initial working directory
/I pathname		Set working directory to directory specified by pathname, which starts from the initial working directory
/P		Set current working directory to previous environment's directory (no arguments allowed)

Argument Switches

None

Examples

```
) DIRECTORY↓
:UDD:USER
) DIRECTORY BETA↓
) DIRECTORY↓
:UDD:USER:BETA
) DIRECTORY/I GAMMA:DELTA↓
) DIRECTORY↓
:UDD:USER:GAMMA:DELTA
) DIRECTORY/I↓
) DIRECTORY↓
:UDD:USER
)
```

First, display the working directory's pathname. Next, make BETA the working directory and display its pathname. Then, using the /I switch and a pathname, make DELTA the working directory. Finally, use 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

/P	Use previous environment's working directory
/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

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↓
)
```

Disconnect the user from the server process whose ID is 12.

DISMOUNT *Command*

Request operator to dismount a tape.

Format

DISMOUNT linkname *[message]*

This command requests the operator to dismount a tape. Specify the same linkname you used to mount the tape.

NOTE: If you issue a DISMOUNT command for a magnetic tape, the system will rewind the tape and restore the drive's ACL to what it was before the mount.

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)
    .
    .
    .
) DISMOUNT MYTAPE PLEASE SEND &)
&)TAPE TO LIBRARY.)
)
```

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. ASCII characters that cannot be printed as text (those whose octal value is less than 40 and greater than 176_8) are replaced by a period(.). Repeated identical lines are indicated by ****.

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 @MTA0 @MTB10 will copy all the files from @MTA0 to @MTB10)
/APPEND	Append the output to the destination file if it exists. The default is to first delete the destination file and then recreate it

/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 zero 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 on the input file and go to the next end-of-file
/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 only the numeric value of the input file, 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 MTBs. The following are valid values for d:

Value	Density
800	800 BPI
1600	1600 BPI
ADM	Automatic Density Matching

/TEXTONLY	List only the text value of the input file, 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↵
```

Produce an octal value and ASCII listing of the file DATA76 on the line printer.

```
) XEQ DISPLAY/ALL/HEXADECIMAL @MTA0↵
```

Produce a hexadecimal and ASCII listing of all the files on MTA0.

```
) XEQ DISPLAY/CONVERT/NOLIST @MTA0 DFILE↵
```

Create 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
/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 dumped files on @OUTPUT
/FLAT	Do not maintain tree structure; dump all files from the specified directories as one directory
/NACL	Dump files without ACLs (later when you LOAD the files, they will be given default ACLs)
/RETAIN=days	Set the retention period of a labelled tape to days. This switch applies only to labelled tape. It will have no effect on other kinds of dump tapes.
/DENSITY=mode	Control the magnetic tape density of your dump tape. Use this command with MTB, model 6026 tape drives only. The following are your mode options: **MODE DENSITY** 800 800 BPI 1600 1600 BPI ADM Automatic Density Matching
/BUFFERSIZE=bytes	Blocks dumped to the tape will have length of bytes
/BEFORE/TLM=date:time	Dump only the files modified before the specified time, date, or date:time
/BEFORE/TLM=time	Dump only the files modified today before time. Time is in the form hh:mm:ss
/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. Date:time is in the form dd-mmm-yy:hh:mm:ss
/BEFORE/TLA=	Dump only the files accessed before the specified time, date, or date:time. See /BEFORE/TLM=date:time for format
/AFTER/TLM=	Dump only the files modified after the specified time, date, or date:time. See /BEFORE/TLM=date:time for format

/AFTER/TLA= Dump only the files accessed after the specified time, date, or date:time. See /BEFORE/TLM=date:time for format

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.

/TYPE=type Select all files of the specified type. Types are provided in Table 2-3. type can be in the following forms:

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; for example

/TYPE=64-68/TYPE=\66

selects types 64, 65, 67 and 68

Argument Switches

None

Examples

```
) DUMP/V/FLAT/NACL FILE6.DUMP :UTIL:+.PR↵
19-JUN-81      10:19:50
   DISPLAY.PR
   SPEED.PR
   EXEC.PR
   XLPT.PR
   LINK.PR
   SED.PR
)
```

Dump all program files in the utilities directory, without their ACLs, to disk file FILE6.DUMP; verify dumped files.

```
) DUMP/AFTER/TLM=4-JUL-80:11:03:42 @MTA0:3↵
)
```

Dump all files in the working directory that were last modified after 11:03:42 a.m. on July 4, 1980 to file 3 of the tape mounted on @MTA0.

!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

Examples

```
) 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 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

Examples

```
) 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 any extension.

Macroname Switches

None

Argument Switches

None

Examples

```
) 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

Examples

```
) [!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 *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 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

Examples

```
) 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 WRITE command.

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

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

Examples

```
) 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 argument$_1$, argument$_2$]

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

Examples

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 access 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
/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
/I	Create input for programname from @INPUT. The last line must contain a single)
/M	Create input for programname from the macro body. The last line of the macro body must contain a single)
/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.

Examples

```
) EXECUTE MYSORT↵
```

Run a program named MYSORT.

```
) EXECUTE/S PROG1↵
```

Run a program named PROG1 and place 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

Examples

[!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

FCU *Utility*

Set nonstandard forms parameters for files to be printed.

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.

Format

XEQ FCU

This calls the Forms Control Utility. The FCU displays the following message on your terminal:

AOS FORMS CONTROL UTILITY 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 wish to use the L command, you must have previously set 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 their dependent parameters default values. If you change the line length, for example, the FCU sets default tab stops.

In all, there are ten 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.

   NOTE: At print time, this number must be less than or equal to the line length of the form in the printer. If you type a NEW LINE character, you are choosing the default of 80 characters per line.

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. So 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 wish 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 FORM LENGTH query).

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 (or 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 (e.g., 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.

FCU (continued)

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 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.

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 3.11      26-NOV-80      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-80      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 calls 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.

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 the octal value 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
000014	------	*053127*
000015	------	*054131*
000016	------	*055012*

```
)
```

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 suffix 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 zero with as short a command line as possible. In addition, using short directory names will permit more filenames to be handled successfully.

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.

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

Use the following command switches to return other 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.

FILESTATUS (continued)

/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

/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.

/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

/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.

/BEFORE/TLM= — List only those files that were last modified before the specified time, date, or date:time

/BEFORE/TLM=time — List only those files that were last modified before time today. time is in the form hh:mm:ss

/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**. **date:time** is in the form **dd-mmm-yy:hh:mm:ss**

/AFTER/TLM= — List only those files that were last modified after the specified **time**, **date**, or **date:time**. See /BEFORE/TLM= **date:time** for format

/BEFORE/TLA= — List only those files that were last accessed before the specified **time**, **date**, or **date:time**. See /BEFORE/TLM= **date:time** for format

/AFTER/TLA= — List only those files that were last accessed after the specified **time**, **date**, or **date:time**. See /BEFORE/TLM= **date:time** for format

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

/TYPE=type — Select all files of the specified **type**. Types are provided in Table 2-3. **type** can be in the form:

XXX	3-letter mnemonic
n	decimal number (0-255)
m-n	decimal numbers which define a range of file types
\n	decimal number to exlude
\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; for example:

/TYPE=64-68/TYPE=\66

selects types **64**, **65**, **67**, and **68**.

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

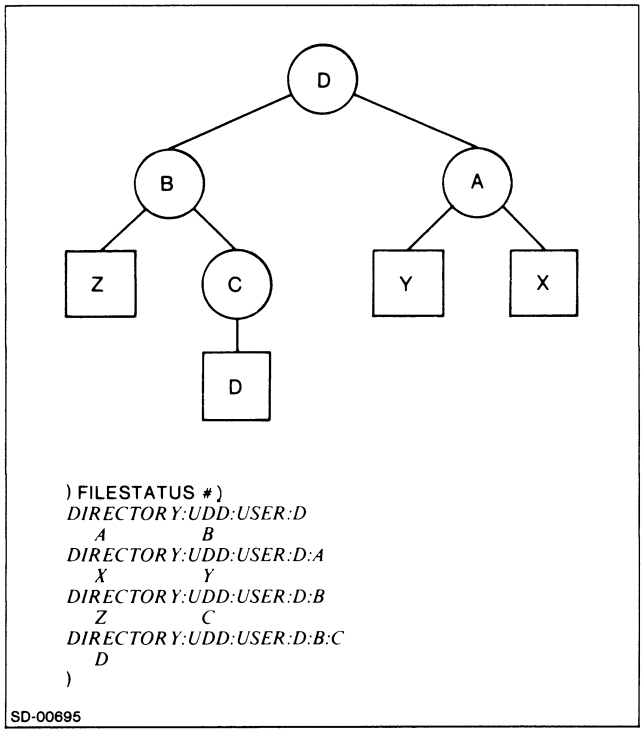
Example

```
) FILESTATUS/ASSORTMENT/SORT
DIRECTORY :UDD:USER

DDAY.DOC  TXT  6-JUN-80     6:00:00  30000
DIR1      DIR  8-DEC-79    14:39:12 123456
LINK2     LNK  :UDD:USERA
PROG.PR   PRG  24-NOV-80    20:44:26   1324

)
```

Display 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.



List the filenames in the tree subordinate to working directory D shown in the accompanying figure.

FORT4 *Utility*

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 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 CLI 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.

Examples

```
) FORT4/B MYPROG↵
```

Compile 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↵
```

Compile separately the four modules that make up the complete program. Then use the Bind utility to load the program. You must load the FORTRAN IV libraries with the program.

F5 *Utility*

Compile a FORTRAN 5 source file.

Format

```
F5 source-pathname
```

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
/D or /DEBUG	Debug. Compile code that allows the long form error traceback routine to output line numbers. Do not use this switch when compiling the final version of your program

F5 (continued)

/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 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

Examples

```
) F5 MYPROG↵
```

Compile 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↵
```

Compile either PROG or PROG.FR. Generate a listing file, PROG.LS, and do not include lines from INCLUDE files in the source listing. Suppress all error messages.

```
) F5 MAIN↵
) F5 SUB↵
) F5 XFUN↵
) F5 XSUB↵
) F5LD MAIN SUB XFUN XSUB↵
```

Compile separately the four modules that make up the complete program. Then use 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 *Utility*

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 CLI 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↵
```

Build compiled FORTRAN 5 module MYPROG.OB into an executable program named NEWPROG.PR. In addition, write a listing to the file named NEWPROG.LM.

F77 *Utility*

Compile a FORTRAN 77 source file.

Format

For AOS:

F77 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 then for a file named pathname.

The F77 macro supplied with AOS automatically includes the /INTEGER=2 and the /LOGICAL=2 switches. You can override the automatic value by explicitly appending the switch to the command and specifying the value 4. 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 any such automatic values.

For a complete description of the FORTRAN 77 programming language and the CLI F77 command line, see the *FORTRAN 77 Reference Manual* (093-000162).

F77 Switches

/CARDFORMAT	Impose card format. Read only the first 72 characters of each line, and pad lines with fewer than 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 errors to the file specified by pathname, rather than to @OUTPUT
/INTEGER=2 or 4	Set the default integer length for this compilation to the specified number of bytes. The F77 macro supplied with AOS automatically contains the switch /INTEGER=2, but you can explicitly specify /INTEGER=4 to override the automatic value
/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 entity length to the specified number of bytes. The F77 macro supplied with AOS automatically contains the switch /LOGICAL=2, but you can explicitly specify /LOGICAL=4 to override the automatic value

/N	Scan the source file as usual, but do not produce an object file. This switch is useful for a quick syntax check
/NOMAP	(AOS/VS only) Do not include a storage map of program entities as part of the listing file
/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 in conjunction 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
/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↵
```

Compile either MYPROG.F77 or MYPROG, depending on whether the source filename has the .F77 extension. The compiler will describe 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↵
```

Compile either PROG or PROG.F77. Generate code for subscript and substring checking. Generate a listing file with alphabetical cross references, and send the listing to the line printer.

```
) F77/OPTIMIZE PROG↵
```

Compile either PROG or PROG.F77. By default, optimization is set to 3, the highest level.

F77LINK *Utility*

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.

Neither the F77LINK macroname nor the switches can be abbreviated.

Any LINK switch can be used as an F77LINK switch. Switches having specific pertinence to F77LINK are given below.

For a complete description of the FORTRAN 77 programming language and the CLI F77LINK command line, see the *FORTRAN 77 Reference Manual* (093-000162).

F77LINK Switches

/CIS	(AOS only) Use the commercial instruction set (CMP, CMV, and FINT). This will allow 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
/PRECONNECTIONS =NONE	Sever all preconnections
/PRECONNECTIONS =*	Use only the preconnections to units 5 and 6
/PRECONNECTIONS =pathname	Use the user-defined preconnections in the .OB file specified by pathname
/ROUND	(AOS/VS only) Direct the ECLIPSE MV/8000™ floating-point unit to use rounding for floating-point values. /ROUND is the default; /TRUNCATE is an option
/TRUNCATE	(AOS/VS only) Direct the ECLIPSE MV/8000™ floating-point unit to truncate floating-point values

Argument Switches

None.

Example

```
) F77LINK MYPROG↵
```

Build compiled FORTRAN 77 module MYPROG.OB into an executable program, MYPROG.PR.

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. However, some of these information files are longer than your screen. If you display such a file, you can freeze the display by typing CTRL-S. You can then read the screen at your leisure and, when you want to resume display, type CTRL-Q.

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
/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 /VOLID= /READONLY

FOR MORE HELP TYPE
HELP/V MOUNT
)
```

!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

Examples

```
) 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

Argument Switches

None

Examples

```
) 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

Examples

```
) 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.../*

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

INITIALIZE (continued)

Examples

```
) 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*

Prepare a magnetic tape with a volume label.

Format

```
XEQ LABEL device-name volid
```

In order to use a labelled tape on AOS or AOS/VS, you must first use the Label utility to prepare the tape. This utility writes a basic set of labels for a null file. The label set may contain a volume label, a header label, and a trailer label. See Chapter 2 for more information on labeled tapes.

device-name is the name of the tape (e.g, @MTA0 or @MTA1). Note that **device-name** should not be a labeled tape device (e.g., @LMT).

volid is an identifier from one to six characters long. You will reference the tape later by this volume identifier.

LABEL Switches

/I	Create IBM labels. If you do not include this switch, the utility will create ANSI standard labels. The type of labels written on the tape at this point will dictate all further use of the tape until you relabel it. In other words, if the utility writes ANSI labels now, all future files read from or written to this volume will be treated as ANSI-labelled files
/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 user volume label(s) after the volume label. The **string** cannot be longer than 76 characters. The system writes one user volume label for each occurrence of this switch. You can specify as many as nine user volume labels.

/OWNER=string — Write string in the volume label's owner field. If you're writing ANSI labels, the maximum length of string is 14 characters. If you're writing IBM labels, the maximum length of string is 10 characters.

Example

```
) XEQ LABEL @MTA0 100000↵
```

Once the tape has been labeled, you can reference it by appending the volid and filename; e.g.,

```
@LMT:100000:filename
```

LMT is the mnemonic for labeled mag tape; 100000 is the volume identifier; filename is the filename you wish to reference.

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).

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

Examples

```
) LEVEL↵
LEVEL 0
) PUSH↵
) ↵
) LEVEL↵
LEVEL 1
)
```

Display current level. Push a level and display 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

Examples

```
) 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, evaluate [!LEVEL] and write the current environment level number. Then, after pushing two levels, evaluate and write [!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 the 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

/ALPHA	List all symbols, sorted alphabetically, and write the list to the list file. /L must accompany this switch
/BUILDSYS	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
/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
/HEX	Convert octal output values to hexadecimal
/KTOP=integer	Set the output program file's address space to integer number of 1K pages

LINK (continued)

/L	Write a listing to the current list file
/L=pathname	Write a listing to the file specified by pathname
/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
/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
/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
/OVER	Suppress the overwrite error message when overwriting occurs
/PRSYM	Place an RDOS-style radix-50 symbol table in the program file
/REV=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
/SRES=integer	Reserve integer number of pages of reserved pages for ?SPAGE I/O
/STACK=integer	Set the default stack size to integer number of words
/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
/V	List the input file pathnames on the listing
/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

/ALIGN=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
/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
/MULT=integer	Give the overlay area integer number of "basic" areas. You may append this switch to the left overlay delimiter (!*)
/OVER	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 (!*)
/ZR /UC /UD /SC /SD = ZR UC UD SC SD integer	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 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

Examples

```
) XEQ LINK/L=@LPT/OVER/ULAST=UC &↓
&) /NSLS/O=MY_PROG ALPHA BETA↓
```

Link two modules, ALPHA and BETA, and name 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 *!↓
```

Bind 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
/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
/K		Set list file to null (no arguments allowed).
/G		Set list file to the generic @LIST file (no arguments allowed). See Chapter 2 for generic filenames
/P		Set list file to the previous environment's list file (no arguments allowed)

Argument Switches

None

Examples

```
) LISTFILE↓
@LIST
) LISTFILE :UDD:PHIL:LIST
)
```

First, display current list file which is the generic @LIST file. Then, set 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

Examples

```
) 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, evaluate [!LISTFILE] and write the current list file, which is the generic @LIST. Then change environment, and set a new list file for the new environment. Evaluate and write [!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 #.

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

/DELETE Delete any existing non-directory file with the same name as a file in the dump file and load the file from the new dump file

/FLAT Do not maintain tree structure; load all files into the working directory

/NACL Give loaded files the default ACL

/N Do not load files. Print dumped files' date and filenames only

/RECENT Do not load if non-directory file on disk is newer than file in dump file

/DENSITY=mode Control the magnetic density of your load tape. Use this command with MTB, model 6026 tape drives only. The following are your mode options:

MODE	DENSITY
800	800 BPI
1600	1600 BPI
ADM	Automatic Density Matching

/V Verify each loaded file on @OUTPUT. When used with /DELETE or /RECENT, the /V switch also verifies each deletion

/BEFORE/TLM= Load only those files that were last modified before the specified time, date, or date:time

/BEFORE/TLM=time Load only those files that were last modified today before time. Time is in the form hh:mm:ss

/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. Date:time is in the form dd-mmm-yy:hh:mm:ss

/AFTER/TLM= Load only those files that were last modified after the specified time, date, or date:time. See /BEFORE/TLM=date:time for format

/BEFORE/TLA= Load only those files that were last accessed before the specified time, date, or date:time. See /BEFORE/TLM=date:time for format

/AFTER/TLA= Load only those files that were last accessed after the specified time, date, or date:time. See /BEFORE/TLM=date:time for format

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.

/BUFFERSIZE=bytes The maximum blocksize of the tape is bytes

/TYPE=type Select all files of the specified type. Types are provided in Table 2-3. 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; for example

/TYPE=64-68/TYPE=\66

selects types 64, 65, 67, and 68.

Argument Switches

None

Examples

```
) LOAD/D/V @MTA0:0↓
DELETED MYFILE.SR_
MYFILE.SR
MYFILE.PR
ZFILE.CLI
DELETED OBS.SR
OBS.SR
)
```

Load all files that are contained in the first file of the magnetic tape mounted on unit 0 into the working directory and list their names on the terminal. If a file in the working directory has the same name as a file in the dump file, delete the file in the working directory and load the file from the dump file. Also, verify all such deletions.

```
) LOAD/RECENT @MTA0:1 +.SR↓
)
```

Load into the working directory each file from @MTA0:1 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
/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
/K		Set the log file to null (no arguments allowed)
/P		Set current log file to the previous environment's log file.
/V		Verify the log file to @OUTPUT

Argument Switches

None

Examples

```
) 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 *Utility*

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.

Two sets of switches are given below, one for AOS and one for AOS/VS. Only a few switches are common to both sets.

MASM Switches (AOS only)

/8	Use the first eight characters of symbols, not just the first five characters
/B=pathname	Name the object file pathname.OB, instead of the name of the first source file
/E	Do not write Pass 2 error messages to @OUTPUT, unless there is no listing file. Some Pass 1 error messages are automatically written to @OUTPUT
/E=pathname	Write the error messages to pathname rather than to @OUTPUT

MASM (continued)

/F	Generate or suppress a form feed as necessary to produce an even number of listing pages. By default, a form feed is generated after each page
/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
/M	Flag redefinition of permanent symbols as mutiple-definition errors
/N	Do not produce an object file
/O	Override all listing suppression controls
/P	Add semipermanent symbols to the cross-reference
/PS=pathname	Use pathname.PS as the .PS file for MASM, instead of MASM.PS
/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
/U	Include user symbols in the object output

Argument Switches (AOS only)

/S	Skip this file on Pass 2 of the assembly

MASM Switches (AOS/VS only)

/CPL=integer	Place integer characters on each line of the various output listings, where integer is in the range from 80 to 136
/E=pathname	Write error messages to pathname, rather than to @OUTPUT
/FF	Generate or suppress a form feed character as necessary to produce an even number of listing pages. /L must accompany this switch
/L	Write a listing to the current list file
/L=pathname	Write a listing to the file specified by pathname
/LOCAL	Include user symbols in the object file
/LPP=integer	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
/MAKEPS	Skip Pass 2, do not produce an object file, and save the temporary symbol table in MASM.PS
/MULT	Flag symbol redefinition statements as errors
/N	Do not generate an object file
/NOPS	Do not use a permanent symbol table to resolve symbols in the source module
/O=pathname	Name the object file pathname.OB, rather than the name of the first source file

/PS=pathname	Use pathname, rather than MASM.PS, as the permanent symbol table for the current assembly
/STATISTICS	Include assembly statistics in the listing. /L must accompany this switch
/SYMBOL=integer	Use the first integer characters to resolve symbols. /MAKEPS or /NOPS must accompany this switch
/ULC	Distinguish between upper- and lower-case letters
/XPAND	List all source lines, regardless of listing suppression directives appearing in the source. /L must accompany this switch
/XREF=NONE or USERSYMBOLS or ALL	Include symbols of the specified type(s) in the cross-reference listing. /L must accompany this switch. By default, the macroassembler includes user symbols in the cross-reference

Argument Switches (AOS/VS only)

/PASS1	Do not process this source file on assembly Pass 2

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 *Utility*

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. 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
/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
/D		Accept arguments as decimal integers; default is octal integers

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 filename1 filename2

This utility runs on both AOS and AOS/VS systems and produces an absolute binary file which will run on a stand-alone system. Filename1 is a LINK save file that you want to input to MKABS. Filename2 is a file you designate to receive output. MKABS outputs filename1 to filename2 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, 16_8 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 16_8 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.

MOUNT *Command*

Mount a tape.

Format

```
MOUNT linkname message
```

Linkname is a filename that you create for the device. Message is the text displayed on the operator's terminal.

This command requests the operator to mount a reel of magnetic tape. After the operator mounts the tape, the system creates a link named linkname in your initial working directory. (MOUNT creates a link for both labeled and unlabeled tapes.) Linkname contains a pathname to the actual device on which the volume was mounted. In subsequent commands, you refer to the physical device by linkname.

After you enter this command, the system locks your terminal until the operator responds.

You need not use the CLI MOUNT command to explicitly mount a volume of labeled tape on a tape drive; simply reference the tape's filename and it will be implicitly 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

MOUNT (continued)

/VOLID=volid

Signal that the MOUNT command refers to a labeled tape or set of labeled tapes. If you include this switch, the EXEC will instruct the operator to mount the specified volid and will create a link for you: :UDD:username:linkname LNK @LMT:volid The EXEC creates a link with the name linkname in directory :UDD:username. This link resolves to @LMT:volid when you issue ?OPEN, ?READ, ?WRITE, or ?CLOSE calls for this volume (see the AOS or AOS/VS programmer's manual for system call explanations). You may substitute linkname for @LMT:volid in the pathname you provide.

You must include the VOLID= switch for each volume in the set in the order in which the system will use the volumes. The EXEC will automatically tell the operator to change volumes as they're needed and the system will verify that the operator has mounted the correct volume each time

/READONLY

Signal that all tapes mounted with this switch are to have ACLs set to username,RE. Request that the write ring be removed from all tapes mounted with this switch.

Argument Switches

None

Examples

```
) MOUNT TAPE1 PLEASE MOUNT TAPE NO Z280↵
) DUMP TAPE1:0 +.PR↵
)
```

First, request the operator to mount a tape numbered Z280 and direct the system to create a link for it named TAPE1. Then dump all of the files with the extension .PR in the working directory to TAPE1.

```
) MOUNT/VOLID=FIRST MYFILE PLEASE &↵
&)REMOVE ENABLE RING↵
)
```

Have EXEC create the link named MYFILE in directory :UDD:username. This link resolves to @LMT:FIRST when you issue an appropriate system call or CLI command. You may substitute MYFILE for @LMT:FIRST in the pathname you provide.

```
) MOUNT/READONLY MYFILE HI DENSITY PLEASE↵
```

Request the operator to mount a tape without a write ring. The tape is therefore read-only.

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 *source files* must be inferior to the working directory. However, you can move any file in any directory with the MOVE/FLAT command which moves the *sourcefiles* to the dest-dir, but does not maintain the tree structure (see Examples). You may use templates for the *sourcefile* argument(s). If you supply no *sourcefile,* the template # is assumed.

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

/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

/BUFFERSIZE=n — Read the *sourcefile(s)* into a buffer of size n bytes

/DELETE — Delete the non-directory file in the dest-dir if it has the same filename as a *source file*

/FLAT — Do not maintain tree structure, move all *source files* into dest-dir (see Examples)

/NACL — Give the new files the default ACL. See the DEFACL command for a discussion of default ACLs

/RECENT — If there is a non-directory 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

/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

/BEFORE/TLM= — Move only those files that were last modified before the specified time, date, or date:time

/BEFORE/TLM=time — Move only those files that were last modified today before time. Time is in the form hh:mm:ss

/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 form dd-mmm-yy:hh:mm:ss

/AFTER/TLM= — Move only those files that were last modified after the specified time, date, or date:time. See /BEFORE/TLM=date:time for format

/BEFORE/TLA= — Move only those files that were last accessed before the specified time, date, or date:time. See /BEFORE/TLM=date:time for format

/AFTER/TLA= — Move only those files that were last accessed after the specified time, date, or date:time. See /BEFORE/TLM=date:time for format

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.

/TYPE=type

Select all files of the specified type. Types are provided in Table 2-3. 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. For example:

/TYPE=64-68/TYPE=\66

selects types 64, 65, 67 and 68.

Argument Switches

None

Examples

```
) MOVE ↑DIR1↓
)
```

Move 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↓
)
```

Move 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 +↓
)
```

Move 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↓
)
```

Move each file ending with the extension .SR which was last modified on or after April Fool's Day of 1984 at 12:30 p.m. and which 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
)
```

Move 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, delete the file in the destination directory. Also, verify the operation by listing the name of each file deleted or 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 reference 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 include 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
/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)
```

Invoke 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

Examples

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, for the same reason as in the !EQUAL pseudo-macro.

!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 *Pseudo-Macro*

Expand to ON or OFF depending on whether the 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.

Macroname Switches

None

Argument Switches

None

Example:

Given a macro including:

```
[!EQUAL, [!OPERATOR],ON]
QBATCH FILE1 FILE2
[!ELSE]
WRITE TRY AGAIN LATER
[!END]
```

!OPERATOR is most useful when you want to change macro behavior based on the presence of an operator. In the example above, 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

Examples

```
) PATHNAME =↓
:UDD:USER:BETA
)
```

Display the pathname of =, the working directory.

```
) PATHNAME TEST↓
:UDD:USER:BETA:TEST
```

Display 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

Examples

```
) CREATE/LINK NIM [!PATHNAME NIM.PR]↓
)
```

Create 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.

Pause 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↓
)
```

Delay 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 *AOS Operator's Guide* (093-000194) and the *AOS/VS Operator's Guide* (093-000244).

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

PERFORMANCE (continued)

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.

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

Examples

```
) PERMANENCE ZONIS↵
OFF
) PERMANENCE ZONIS ON↵
) PERMANENCE/V ZONIS↵
ZONIS          ON
)
```

First, display the PERMANENCE status of file ZONIS. Second, set PERMANENCE for the file to ON. Last, display and verify 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 *Utility*

Compile a PL/I source file.

Format

XEQ PL1 source-pathname

PL/I is a high-level language based on ANSI standard PL/I. You compile a PL/I source file by using the PL1 utility. PL/I source files have the extension .PL1. The compiler first searches for pathname.PL1, and then for pathname. The compiler produces an object file named source-pathname.OB.

Under AOS/VS, PL/I accepts abbreviations for switch names, provided the abbreviation is unique. Under AOS, however, switch names cannot be abbreviated.

For detailed information on the PL/I programming language, see *Plain PL/I (A PL/I Primer)* (093-000216), the *PL/I Reference Manual (AOS)* (093-000204), and the *PL/I Reference Manual (AOS/VS)* (093-000270).

PL/I Switches (AOS only)

/CODE	Print a generated code listing on the list file. /L must accompany this switch
/DEBUG	Output symbol and line information for use by SWAT™ debugger
/E=pathname	Write error messages to pathname instead of @OUTPUT
/ERRORCOUNT =integer	Terminate compilation after integer errors. If you do not include this switch, the default is 100 errors
/L	Write a listing to the current list file. A listing contains line-numbered source text, storage map (unless /NOMAP switch is used), compilation error messages, and compilation statistics. For more information, use the appropriate switch in combination with /L: for example, /XREF includes symbol cross-reference
/L=pathname	Write a listing to pathname

PL1 (continued)

/LINEID	Generate code to keep track of source line numbers at execution time
/N	Do not produce an object file
/NEST	Print nesting level of blocks and groups on source listing
/NOINCLUDES	Suppress listing of all %INCLUDE files
/NOLEF	Do not generate LEF instructions
/NOMAP	Suppress listing of storage map
/NOPROCID	Do not save procedure names in the static data area
/NOWARNINGS	Suppress severity 1 error messages
/O=pathname	Write object file to pathname.OB, or to pathname if it already ends in .OB
/OPT [=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
/PARAMS=pathname	Include parameter file pathname in the present compilation. The parameter file must have been previously created by compiling with the /SAVEPARAMS= switch
/SAVEPARAMS [=pathname]	Compile only %REPLACE statements, and save the result in pathname for later use with the /PARAMS= switch. If you do not specify a pathname, the compiler names the file source-pathname.PAR
/STAT	Write compilation statistics to @OUTPUT
/SUB	Compile code into program to check for out-of-bounds subscripts and arguments to SUBSTR
/TMPDIR=string	Add string as a prefix to the beginning of all temporary filenames
/XREF	Append a cross-reference table to the listing file. /L must accompany this switch

PL/I Switches (AOS/VS only)

/CODE	Print a generated code listing on the list file. /CODE overrides /CODEMAP. /L must accompany this switch
/CODEMAP	Print a code offset map. If /CODE is also specified, the compiler ignores /CODEMAP. /L must accompany this switch
/DEBUG	Output symbol and line information for use by SWAT™ debugger
/DECLARES =pathname	Use pathname, created by a previous compilation with /SAVEDECLARES=, as the initial symbol table
/E=pathname	Write error messages to pathname instead of @OUTPUT
/ERRORCOUNT =integer	Terminate compilation after integer errors. If you do not include this switch, the default is 100 errors
/INCLUDES	Do not suppress listing of %INCLUDE files. This is the default

/L	Write a listing to the current list file. A listing contains line-numbered source text, storage map (unless /NOMAP switch is used), and compilation error messages. For more information, use the appropriate switch in combination with /L: for example, /XREF includes symbol cross-reference
/L=pathname	Write a listing to pathname
/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. This switch 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	Do not produce an object file
/NESTLEVEL	Print nesting level of blocks and groups on source listing
/NOCODE	Do not print a generated code listing on the listing file. This is the default
/NOCODEMAP	Do not print a code offset map. This is the default
/NODEBUG	Do not output symbol and line information for SWAT. This is the default
/NOINCLUDES	Suppress listing of all %INCLUDE files
/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 listing of storage map
/NOMAPCASE	Do not translate identifiers into uppercase. This is the default
/NONESTLEVEL	Do not include block and group nesting level numbers in listing. This is the default
/NOPROCID	Do not generate code to keep track of procedure names at runtime. This is the default
/NOREPLACES	Do not flag lines containing %REPLACE names. This is the default
/NOSTATISTICS	Do not print compilation statistics. This is the default
/NOSUBCHECK	Do not generate code to check out-of-bound subscripts and arguments to SUBSTR at runtime. This is the default
/NOWARNINGS	Suppress severity 1 error messages
/NOXREF	Do not generate a symbol cross-reference table. This is the default
/O=pathname	Write object file to pathname.OB, or to pathname if it already ends in .OB
/OPT [=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 all procedure names at runtime. The default on-unit reports the procedure in which a runtime error occurs
/REPLACES	Flag listing lines containing %REPLACE names

PL1 (continued)

/SAVEDECLARES =pathname	Save symbol table in pathname. The source file may contain only DECLAREs and %REPLACEs
/STATISTICS	Write compilation statistics to the listing file and @OUTPUT
/SUBCHECK	Compile code into program to check for out-of-bounds subscripts and arguments to SUBSTR
/SYMLIB =pathname	Use pathname, created by a previous compilation with /SAVEDECLARES=, as a symbol "library"
/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	Append a cross-reference table to the listing file. /L must accompany this switch

Argument Switches

None.

Example

```
) XEQ PL1/L=LFILE MYPROG↵
```

Compile the PL/I source file MYPROG.PL1. The /L=LFILE switch directs the compiler to output the listing to file LFILE.

PL1LINK *Utility*

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 LEVEL, SUPERPROCESS, SUPERUSER, SCREENEDIT, SQUEEZE, CLASS1, CLASS2, VAR0 through VAR9, TRACE, LISTFILE, DATAFILE, LOGFILE, DIRECTORY, SEARCHLIST, DEFACL, STRING, PROMPT, and CHARACTERISTICS. It preserves none of the current settings. If your current LEVEL is 0, the POP command will cause 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

Examples

```
) LEVEL↵
LEVEL 1
) POP/V↵
LEVEL 0
)
```

First display current level, then POP to the previous environment and verify 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
/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
/I		Set the prefix to the initial value, that is, a right parenthesis (no arguments allowed)

Example

```
) PREFIX [!ASCII 207] HELLO↵
HELLO PREFIX/I↵
)
```

First, set 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

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/EB1/ULC/WRP
                        /OFF/SFF/EPI/8BT/SPO
/RAF/RAT/RAC/NAS/OTT/EOL/UCO/LT/FF
/PM/NRM/MOD/TO/TSP/PBN/ESC/FKT/NNL

)
```

First display the current level. Then, display all the previous environment's settings.

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 mode 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.

If you input PRIORITY with no arguments, the system will display the CLI's priority.

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↓
)
```

Set 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 simple /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.

NOTE: 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.

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
/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 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

PROCESS (continued)

/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

/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 simple /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

/LIST — Make the new process's generic @LIST filename the same as the creating process's. The default is no @LIST filename

/LIST=pathname — Pathname is the new process's generic @LIST filename

/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 512MB. For 16-bit programs running under AOS/VS, the default is minimum of top of shared or 64KB

/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
/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
/RESIDENT	Make the new process resident.

NOTE: If you omit the /PREEMPTIBLE and /RESIDENT switches, the process is swappable

/SONS	A son process may create the same number of processes as the creating process, minus one; default is zero
/SONS=number	Make number the maximum number of son processes that the new process can create
/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
/USERNAME=name	Make name the new process's username; the default is the same as creating process's
/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.

Examples

```
) PROCESS/IOC=@CON1 UPDATE↵
PID: 13
)
```

Create 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↵
)
```

Create a swappable son process with the same generic @INPUT, @OUTPUT, and @CONSOLE as this CLI, and block 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 8 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 Examples). You can only use commands that do not require an 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
/K		Set PROMPT to null (no arguments allowed)
/P		Set PROMPT to previous environment's PROMPT (no arguments allowed)

Argument Switches

None

Examples

```
) PROMPT TIME DATE DIRECTORY↵
9:32:16
26-NOV-80
:UDD:USER:PHIL
) PROMPT↵
TIME
DATE
DIRECTORY
9:32:18
26-NOV-80
:UDD:USER:PHIL
) PROMPT/K↵
)
```

First, set 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, display PROMPT; then set 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.

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, create a preemptible son process with program PROGA. (The system assigns the new process a PID of 14.) Then, set 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 LEVEL, SUPERPROCESS, SUPERUSER, SQUEEZE, CLASS1, CLASS2, TRACE, VAR0 through VAR9, LISTFILE, DATAFILE, LOGFILE, SCREENEDIT, DIRECTORY, SEARCHLIST, DEFACL, STRING, PROMPT, and CHARACTERISTICS, 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

PUSH (continued)

Argument Switches

None

Examples

```
) LEVEL↵
LEVEL 0
) PUSH/V↵
LEVEL 1
)
```

First, display the current environment's level; then PUSH a level (thereby saving the current environment) and display 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
/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
/I		Take the contents of the file from subsequent lines of the @INPUT file. You must terminate the input mode with a single right parenthesis), and a NEW LINE (no arguments allowed)
/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)

/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
/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
/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 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
/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	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
/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
/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. The pathname should not be a queuename
/QPRIORITY=n	Give this job priority n (0 <n< 255). n job priority specified in your user profile.

Argument Switches

Use any argument switches appropriate for batch job specified in argument.

QBATCH (continued)

Examples

```
) 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

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

A Unexpired /AFTER switch in effect

B /BINARY switch in effect

C Cancelled by user (QCANCEL command)

D User /DELETE switch in effect

E Held by operator

F Cancelled 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

QDISPLAY (continued)

Argument Switches

None

Examples

) QDISPLAY↵

Display information about all queues.

) QDISPLAY/TYPE=BATCH/TYPE=PRINT↵

Display information about the batch and print queues.

) QDISPLAY/QUEUE=BATCH_INPUT↵

Display information about the queue named BATCH_INPUT.

QFTA *Command*

Place an entry on the FTA queue.

Format

QFTA/DESTINATION=pathname source-pathname

This command queues the file named in source-pathname to the file transfer agent (FTA) queue. Note that this command does not transfer the file, but merely queues it to the transfer agent; so don't delete or modify the file until the system transfers it.

The source-pathname may be local or remote. If it is remote, it must be a complete pathname, beginning from :NET:hostname.

The /DESTINATION=pathname switch is required. The pathname must be complete, beginning from :NET:hostname.

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 sequence number in STRING where you can use it as an argument to commands via the !STRING pseudo-macro

/V	Display the pathname of the queued file
/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 result will not be processed before a certain time. The quest 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
/APPEND	Append the source file to the destination file if it already exists
/CHECKPOINT	Restart the transfer from the most recent checkpoint if a network error occurs during the transfer
/COMPRESS [=integer]	Use data compression algorithm integer. If you use this switch but do not supply an integer, the default value is 0. If you do not use this switch, the system transfers the data without compression NOTE: Currently, only algorithm 0 is available.
/DDELETE	Delete the destination file (if it exists) before the transfer occurs
/DESTINATION =pathname	This switch is required. The pathname may be remote or local, but it must be fully qualified
/HOLD	Hold the entry until you explicitly unhold it with the QUNHOLD command
/NORESTART	Do not restart the transfer if the system fails during the transfer
/NOTIFY	Tell the FTA to send a message to your terminal upon completion of the transfer
/OPERATOR	Do not run this job unless an operator is present
/QPRIORITY=n	Give the job priority n. 1 is the highest priority, and 255 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 rather than to the default queue. The queue type must be FTA
/RECENT	Process the request only if the source file is more recent than the destination file
/RMODE	Use record mode transfer. The default is block mode
/SDELETE	Delete the source file after the transfer completes
/STREAM=integer	Submit the job to stream n, where integer is in the range 0 to 7. The default stream is 0

Argument Switches

None

Example

```
) QFTA/DESTINATION=:NET:HOSTA:UDD:USER &↓
&) :FILEX/VFILE1↓
:UDD:USER:FILE1 QUEUED, SEQ=32, QPRI=127
)
```

Submit FILE1 to the FTA queue, and display the name of the queued file. The destination file is remote, and must be fully qualified with the :NET directory name and the :HOSTA hostname.

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↵
)
```

Hold 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
/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 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

/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 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

/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 which would require special forms

/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

Argument Switches

None

Examples

```
) QPLOT FILE1 FILE2↵
QUEUED, SEQ=651, QPRI=127
QUEUED, SEQ=652, QPRI=127
)
```

Queue FILE1 and FILE2 to a digital plotter output queue.

```
) QPLOT/DELETE FILE3↵
QUEUED, SEQ=653, QPRI=127
)
```

Plot FILE3 then delete FILE3 when it is complete.

```
) QPLOT/COPIES=3/TITLES FILE4↵
QUEUED, SEQ=654, QPRI=127
)
```

Plot 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

/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 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

/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 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 which 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

/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

/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

/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

Argument Switches

None

Examples

```
) QPRINT FILE1 FILE2↵
QUEUED, SEQ=655, QPRI=127
QUEUED, SEQ=656, QPRI=127
)
```

Queue FILE1 and FILE2 to the line printer output queue.

```
) QPRINT/DELETE/FOLDLONGLINES FILE3↵
QUEUED, SEQ=657, QPRI=127
)
```

Print FILE3 folding long lines. Delete FILE3 when it finishes printing.

```
) QPRINT/COPIES=3/PAGES=75/TITLES FILE4↵
QUEUED, SEQ=658, QPRI=127
)
```

Print three copies of FILE4 and produce 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
/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 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
/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 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
/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
/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

Argument Switches

None

Examples

```
) QPUNCH FILE1 FILE2↓
QUEUED, SEQ=659, QPRI=127
QUEUED, SEQ=660, QPRI=127
)
```

Queue FILE1 and FILE2 to the paper tape punch output queue.

```
) QPUNCH/DELETE FILE3↓
QUEUED, SEQ=661, QPRI=127
)
```

Punch FILE3, then delete FILE3 when complete.

```
) QPUNCH/COPIES=3/FEET=75 FILE4↓
QUEUED, SEQ=662, QPRI=127
)
```

Punch three copies of FILE4. Do not punch more than 75 feet of paper tape.

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 QPLOT, QPRINT, or switches for other options you can specify.

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
/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 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

QSUBMIT (continued)

/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 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

/BINARY

Print in binary mode. This switch is valid only for devices which have 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

/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

/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

/QOUTPUT=pathname

Set the generic output file of the batch process to pathname. The pathname should not be a queuename

/QLIST=pathname	Set the generic list file of the batch process to pathname.
/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

Examples

```
) QSUBMIT FILE1 FILE2↵
QUEUED, SEQ=663, QPRI=127
QUEUED, SEQ=664, QPRI=127
)
```

Submit FILE1 and FILE2 to the BATCH_INPUT

```
) QSUBMIT/NORESTART/HOLD FILE3↵
QUEUED, SEQ=665, QPRI=127
)
```

Submit FILE3 to the BATCH_INPUT queue. Hold it until a subsequent QUNHOLD command releases it. Do 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.

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

Argument Switches

None

Example

```
) QHOLD MYJOB↵
      .
      .
      .
) QUNHOLD MYJOB↵
)
```

First hold, and then unhold, 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 which 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

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

/D Load the new file after deleting any file with the same filename.

/N Do not load, just verify filenames.

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.

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

/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.

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

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.

Format

XEQ RDOS PUT/DISK=rdos-diskunit
[/DIR=rdos-subdirectory] [aos-pathname]...[templates]

PUT Switches

/DIR=rdos-subdirectory Put files into the specified RDOS subdirectory.

Index Level Considerations

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.

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

/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.

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.

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 When used with LOAD and GET commands, convert carriage returns to NEW LINES. When used with DUMP and PUT commands, convert NEW LINES to carriage returns.

/N Do not transfer files matching this template.

Examples

```
) XEQ RDOS LOAD @MTA0:0   +.SR/C   +.RB↵
```

Load all the files ending in .SR and .RB from file 0 on MTA0 into the working directory. Convert all carriage returns in the source files (.SRs) to NEW LINEs; do not convert them in the relocatable binary files (.RBs).

```
) XEQ RDOS LOAD/V @MTA0:1 +.SV/N↵
```

Load all files on @MTA0:1 except files with an .SV extension.

```
) XEQ RDOS DUMP/V @MTA0:1 +/C↵
```

Dump all files in the working directory to file 1 of MTA0. Convert all of the source files' NEW LINEs to carriage returns, and list 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↵
```

Copy 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↵
```

Copy 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:-.-↵
```

List 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:+↵
```

List 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↵
```

List the filenames of all files on RDOS disk @DPD5, including the filenames in each subdirectory.

Erroneous Examples

There are several common errors made in the use of 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.

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
)
```

Release an LD named ALPHA that you previously INITIALIZED.

RENAME *Command*

Change a file's name.

Format

RENAME pathname newname

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

Examples

```
) 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 *AOS Operator's Guide* (093-000194) and the *AOS/VS Operator's Guide* (093-000244).

REVISION *Command*

Set or display a program's revision number.

Format

REVISION pathname *[field1[.field2[.field3[.field4]]]]*

You may use templates in the pathname argument. *field1* and *field2* 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 files of type PRG have only two field numbers whereas files of type PRV have four field numbers.

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

Argument Switches

None

Examples

```
) 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.

Examples:

```
) REWIND @MTA0↓
```

Rewind the magnetic tape on unit @MTA0.

```
) MOUNT TAPE10 MOUNT_IT_AGAIN_SAM↓
      .
      .
      .
) REWIND TAPE10↓
)
```

First request the operator to mount a tape and create a link named TAPE10 to that tape. After performing the required operations, rewind TAPE10.

RIC *Utility*

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	Write the compiler-generated object code to the output file. The code follows the source listing (if one is specified) and the numbers of the lines for which DG/RIC generated specific 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
/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=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. /N 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.

Examples

```
) RIC MYPROG↵
```

Compile 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↵
```

Compile MYPROG.RG with debugger and analyzer information in the object file. The created files are named NEWNAME.IC, NEWNAME.PR, and NEWNAME.DL.

```
) RIC/SOURCE/L=PROG.LS MYPROG↵
```

Compile MYPROG.RG, and send a source listing and error messages to the output file PROG.LS.

ROC *Utility*

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 as many buffers as will fit. If you do not use a /BUFFERS switch, 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 (/DSPLYRTN, /INTVAR, and /OPT=2) 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 directory :TEMP, you specify the prefix for these files, that is, :TEMP:

ROC (continued)

/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.

Examples

) ROC MYPROG↵

Compile MYPROG.RG and produce optimized machine code.

) ROC/SOURCE/L=PROG.LS MYPROG↵

Compile MYPROG.RG, and send a source listing and error messages to the output file PROG.LS.

) ROC/P/TRACE/L=PROG.LS MYPROG↵

Compile MYPROG.RG, and send a listing to PROG.LS. Also, generate 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 *Utility*

Compile an RPG II source file.

Format

RPG source-pathname

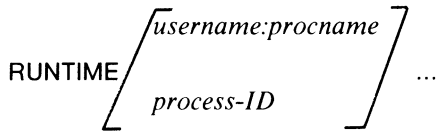
When you invoke the RPG macro, the system actually executes the RIC macro. RPG.CLI is a link to RIC. See the description of RIC.

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



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)

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 Command 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 only if the next two lines following the above series matched.

The maximum number of characters per line for either file is 132, 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

/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
/EOL	The end-of-line character is treated as significant. If you omit this switch, SCOM ignores EOL characters and blank lines
/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 input 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 &). Insertion mode (CTRL-E) is terminated as soon as the number of characters originally input plus the newly inserted ones totals 76. 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.

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
CTRL-I	Insert a tab
CTRL-K	Erase everything to the right of the cursor
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↵
) ANY 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
/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
/Q		Set SQUEEZE to ON for this command
/K		Delete the current search list, if any exists (no arguments allowed)
/P		Set search list to previous environment's search list (no arguments allowed)

Argument Switches

None

Examples

```
) SEARCHLIST↵
:PER,:UTIL,:
) PUSH↵
) SEARCHLIST :UDD:HENRY,:PER,:UTIL,:↵
) SEARCHLIST↵
:UDD:HENRY,:PER,:UTIL,:
) SEARCHLIST/P↵
) SEARCHLIST↵
:PER,:UTIL,:
)
```

Display the current search list and then push a level. Change the current search list and display it. Then set the search list to its original value and display 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]↵
)
```

Evaluate the !SEARCHLIST pseudo-macro, then set 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=pathname	Store the .ED file, which contains formatting settings, in the directory specified by pathname
/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=pathname	Store all temporary SED files in the directory specified by pathname

Argument Switches

None.

Example

```
) XEQ SED/ED=NEAT/WORK=:FIX_HEAD REPORT↵
```

Invoke SED to edit the file REPORT. Place 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   }  message
     { consolename         }
```

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, or control characters in your messages. Commas and tabs become spaces when the system sends your message. 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 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
/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
/I	Send each of the following input lines as a separate message until you reach a line containing a single)
/M	Send each line of the current macro file as a separate message. You must end the macro sequence with a single)

Argument Switches

None

Examples

```
) SEND 2 PLEASE BRING UP LPT1↓
)
```

Send the message to the system operator.

```
) SEND @CON- SYSTEM WILL SHUT DOWN &↓
&)AT MIDNIGHT.↓
)
```

Send 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.↓
)
```

Reply 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 zero.

Macroname Switches

None

Argument Switches

None

Examples

```
) 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↓
```

Create 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 *Utility*

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
```

Invoke 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 pecified 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 control-point-directory or logical-disk name

Argument Switches

None

Examples

```
) SPACE/V↵
= MAX 370889, CUR 304097, REM 66792
)
```

Display disk space in =, the working directory.

```
) SPACE :↵
MAX 37000, CUR 18000, REM 19000
)
```

Display 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

/D	Display text automatically. If you include both /D and /I, /D is ignored
/I=pathname	Take the SPEED commands from the file specified in pathname, rather than from @INPUT, that is, the keyboard

Argument Switches

None.

Example

) XEQ SPEED/I=PROCESS_REP.SCF REPORT1↓

Invoke 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
/Q		Set SQUEEZE to ON for this command
/P		Set the current SQUEEZE mode to the previous environment's SQUEEZE mode (no arguments allowed)

Argument Switches

None

Examples

```
) SQUEEZE↵
OFF
) SQUEEZE ON↵
) SQUEEZE↵
ON
)
```

Display the current SQUEEZE setting, then set SQUEEZE to ON, and finally display the new SQUEEZE setting.

STRING *Command*

Set or display STRING setting.

Format

STRING *[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
/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
/K		Set STRING to null (no arguments allowed)
/P		Set STRING to previous environment's STRING (no arguments allowed)

Argument Switches

None

Examples

```
) STRING↵
THIS,IS,A,STRING
) STRING NEW ONE↵
) STRING↵
NEW,ONE
) STRING/K↵
) STRING↵
)
```

First, display STRING, then set it to a new string. Kill the current STRING, and display 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

Examples

```
) 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
/Q	Set SQUEEZE to ON for this command

/P	Set the current SUPERPROCESS mode to previous environment's SUPERPROCESS mode (no arguments allowed)

Arguments Switches

None

Examples

```
) SUPERPROCESS↓
OFF
) SUPERPROCESS ON↓
+)SUPERPROCESS↓
ON
+)
```

First, display the current SUPERPROCESS setting. Second, turn SUPERPROCESS ON. (Note that with SUPERPROCESS turned ON, the CLI outputs the prompt +) in the left margin). Last, display the current SUPERPROCESS setting.

SUPERUSER *Command*

Set or display the SUPERUSER setting.

Format

SUPERUSER [*ON* / *OFF*]

Only privileged processes can set SUPERUSER to *ON*. See Chapter 4 for more information about SUPERUSER.

The CLI precedes each prompt with an asterisk (*) when you have SUPERUSER turned to *ON*. If you enable both SUPERUSER and SUPERPROCESS (see previous 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
/Q	Set SQUEEZE to ON for this command
/P	Set the current SUPERUSER mode to previous environment's SUPERUSER mode (no arguments allowed)

SUPERUSER (continued)

Argument Switches

None

Examples

```
) SUPERUSER↓
OFF
) SUPERUSER ON↓
*)SUPERUSER↓
ON
*)
```

First, display the current SUPERUSER setting. Second, turn SUPERUSER ON. (Note that with SUPERUSER turned ON, the CLI outputs the prompt *) in the left margin). Finally, display 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 writes the information to file 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↓
```

Invoke SWAT to debug program MYPROG. For debugging, use 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]...*

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

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

```
) SYSINFO↓
SYSTEM REVISION: 03.30
SYSTEM MEMORY: 1023 PAGES
MASTER LOGICAL DISK: ROOT.7.25.79
SYSID: R E M U L A C
)
```

This displays current system information.

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
/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
/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
/START		Start system log file
/STOP		Stop system log file

Argument Switches

None

Example

```
) SYSLOG/START↓ )
```

Start 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
/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
/BREAKFILE[=pathname]	Produce a breakfile in the working directory at the time of termination. If you call TERMINATE with the simple /BREAKFILE switch, the system creates a break file with a default name. If you call TERMINATE with /BREAKFILE=pathname, the break file will have the specified name

Argument Switches

None

Example

```
) PROCESS SMITH:PROGA↵
PID 17
      .
      .
      .
) TERMINATE 17↵
)
```

First, create a swappable son process that runs concurrently with the CLI. (The CLI displays the PID of the new process.) Then terminate the process.

TIME *Command*

Set or display the current system time.

Format

TIME *[new-time]*

Only the operator (PID2) can set the time. *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

Examples

```
) TIME↓
19:30:45
) TIME 8 45↓
)
```

Display the system time, then set 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

Examples

```
) 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
/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
/COMMAND	Specify command tracing
/LOG	Send trace output to the log file
/KILL	Turn all tracing modes off (no arguments allowed)
/MACRO	Specify macro tracing
/PREVIOUS	Set trace mode to previous trace mode (no arguments allowed)
/PSEUDO	Specify pseudo-macro tracing
/ON	Turn the following tracing modes on
/OFF	Turn the following tracing modes off

Argument Switches

None

Examples

```
) TRACE/MACRO↵
) TRACE/COMMAND↵
) TRACE↵
***TRACE
/COMMAND/MACRO
)
```

First turn on macro tracing, and then turn on command tracing. Then display 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

Examples

```
) TREE 7↵
PID: 7, FATHER: 4, SONS: 8 12 13
) TREE OP:EXEC↵
PID: 4, FATHER: 2, SONS: 7 9 10 11
)
```

Display PID 7's tree. Then, display EXEC's tree.

TYPE *Command*

Type the contents of a file.

Format

TYPE pathname *[pathname]...*

SQUEEZE mode doesn't compress output from TYPE.

You may use templates for 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
/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 title and record type of the file before typing it. If the record type is fixed, the CLI also displays the record length

Argument Switches

None

TYPE (continued)

Examples

```
) TYPE MYFILE↵
      .
      .
      .
) TYPE/2=ERROR FILE1 FILE2 FILE3↵
      .
      .
      .
)
```

Display MYFILE on the terminal. Then, set CLASS2 exceptional conditions to ERROR and type 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

Examples

```
) WRITE [!UADD 5 6]↵
11
)
```

Add two integers and display the sum.

Given a macro containing:

```
      .
      .
      .
[!ULE,[!UADD,[!SIZE FILE1],[!SIZE FILE2],10000]
      .
      .
      .
```

Evaluate 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 zero.

Macroname Switches

None

Argument Switches

None

Examples

```
) WRITE [!UDIVIDE 10 3]↓
3
)
```

Divide the first integer by the second, and display 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
```

Push a level, prompt for input, and assign the value to a variable. Multiply the amount by .15: that is, multiply the amount by 15, and then divide the product by 100. Output this final quotient.

!UEQ *Pseudo-Macro*

Include input conditionally.

Format

[!UEQ argument$_1$ argument$_2$]

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.

!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.

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

!UEQ (continued)

Argument Switches

None

Examples

Given a macro containing:

```
.
.
.
[!UEQ,[!VAR0],17]
WRITE VAR0 = 17
[!ELSE]
WRITE VAR0 = [!VAR0]
[!END]
```

This macro will write *VAR0 = 17* if the current value of VAR0 is 17. Otherwise, it will write *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$_2$]

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 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.

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

Examples

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 will write the first message if the current value of VAR7 is greater than or equal to 10. Otherwise, it will write 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$_2$]

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 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.

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

!UGT (continued)

Examples

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 will write the first message if the current value of VAR1 is greater than 11. Otherwise, it will write 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$_2$]

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 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.

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

Examples

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 will write the first message if the current value of VAR5 is less than or equal to 2. Otherwise, it will write 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$_2$]

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 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.

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

!ULT (continued)

Examples

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 zero.

Macroname Switches

None

Argument Switches

None

Examples

```
) WRITE [!UMODULO 10 3]↓
1
)
```

Perform the modulus operation on two integers, and display 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%]
```

Evaluate and display the !UDIVIDE pseudo-macro, then evaluate and display the !UMODULO pseudo-macro.

!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

Examples

```
) WRITE [!UMULTIPLY 66777 92]↵
6143484
)
```

Multiply two integers and display 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]
```

Subtract 32 from the Fahrenheit argument. Multiply the difference by 5/9: that is, multiply the amount by 5, and then divide the product by 9. Finally, output 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

UNBLOCK (continued)

Example

```
) PROCESS SMITH:PROGA↓
PID 17
) BLOCK 17↓
      .
      .
      .
) UNBLOCK 17↓
)
```

First, create an inferior swappable process that runs concurrently with the CLI. Block the new process, perform the required operations, then unblock the new process.

!UNE *Pseudo-Macro*

Include input conditionally.

Format

[!UNE argument$_1$ argument$_2$]

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.

!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.

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

Examples

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 will write *VAR0 = n*, where n is the current value of VAR0. Otherwise, it will write *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

Examples

```
) WRITE [!USUBTRACT 17 5]
12
)
```

Subtract one integer from another and display 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, determine if the first argument is greater than the second. If it is, subtract the second from the first and display the result. If the second argument is greater than the first, subtract the first from the second, put a negative sign before the difference, and display 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
/Q		Set SQUEEZE to ON for this command
/P		Set the current value of the variable to the previous environment's value (no arguments allowed)

Example

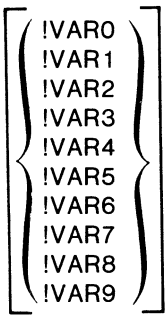
```
) VAR0↓
0
) VAR0 34573↓
) WRITE [!UADD [!VAR0] 2]↓
34575
)
```

First, display the current value of VAR0. Then reset the value. Finally, include 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



The ten 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

Examples

```
) 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, evaluate [!VAR0] and write the current value of VAR0. Then change environment, and give VAR0 a new value. Evaluate and write [!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 *Managing 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

Examples

```
) 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 CLASS2to 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

Examples

```
) 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).

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

XEQ (continued)

/I	Create input for the program from @INPUT. The last line of input must contain a single)
/M	Create input for program from the macro body. The last line of the macro body must contain a single)
/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.

Examples

```
) XEQ LINK OBJ1↵
```

Call the LINK utility; that is, create a subordinate process whose program is the LINK utility. Have LINK produce an executable program file from the object file OBJ1.

```
) XEQ MYPROG 0 1↵
```

Execute a program named MYPROG. Note that MYPROG can get the arguments 0 and 1 for use in whatever way you choose.

```
) XEQ/S PROG2↵
```

Execute PROG2; divert PROG2's termination message (if any) to STRING instead of to @OUTPUT.

End of Chapter

Appendix A
Exceptional Condition Messages

This appendix describes the exceptional condition messages you might receive when you enter an invalid CLI command line.

Errors that occur while you're running a utility are described in the manual for the 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.

MESSAGES

/AFTER OR /BEFORE SWITCH REQUIRED

When you use either the /TLA= or /TLM= switch, you must also use the /AFTER or /BEFORE switch.

ARGUMENT IS NOT A COMMAND

You've used an invalid argument in a PROMPT command.

ARGUMENT IS NOT A UNIQUE ABBREVIATION

You have entered an invalid abbreviation as an argument to the PROMPT command.

ARGUMENT MAY NOT BE A NONIMPLEMENTED COMMAND

An argument in a PROMPT command has not yet been implemented.

ARGUMENT MAY NOT BE A COMMAND REQUIRING ARGUMENT(S)

An argument in a PROMPT command requires arguments and, therefore, it is an invalid argument to PROMPT.

ARGUMENT MAY NOT HAVE SWITCHES

An argument to the PROMPT command has switches and is, therefore, invalid.

CAN'T POP FROM LEVEL 0

You've issued the POP command from LEVEL 0. Level 0 is the highest level.

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 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.

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.

FILE IS NOT A CONTROL POINT DIRECTORY

You've issued the SPACE command on a file that is not a control point directory.

FIXED RECORD LENGTH IS ZERO

You tried to type a file with zero-length fixed length records.

ILLEGAL DECIMAL NUMBER

The argument to this command must be an unsigned, positive, decimal number.

ILLEGAL DECIMAL SWITCH VALUE

ILLEGAL FILE TYPE

You have tried to move or dump a file with type less than 100 (octal).

ILLEGAL FILENAME TEMPLATE

You have entered an invalid filename specification in a command line.

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.

INDECIPHERABLE DUMP FORMAT

You've entered an invalid format for a dumpfile. The dumpfile may not have been created by the CLI DUMP command, or it may be a tape file with a buffer size different from the specified or default size on the LOAD command.

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 TIME FORMAT

You entered an invalid argument to the TIME command or to a /TLA or /TLM command switch.

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.

NO FILES MATCH TEMPLATE

NO MACRO INPUT AVAILABLE

You've used the /M command switch and there is no terminator] in the macro.

NO PREVIOUS VALUE WHEN AT LEVEL 0

You've issued the PREVIOUS command or the /P command switch from LEVEL 0.

NON-UNIQUE QUEUE TYPE ABBREVIATION

NOT A COMMAND OR MACRO

You began a command line with an item which is not a command or macro.

*****NOT ENOUGH MEMORY, RESTARTING CLI*****

The CLI required more memory than you allocated it. 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.

PARENTHESES NOT ALLOWED IN MACRO NAME

You've typed parentheses within a macroname, which is illegal.

PATHNAME MUST START FROM WORKING DIRECTORY

You've specified an invalid pathname in a LOAD, MOVE, or DUMP command.

PATHNAME TOO LONG

Pathnames cannot exceed 128 characters.

PROCESS NUMBER TERMINATED BY CONSOLE INTERRUPT

You've aborted the process in control of the console with a CTRL-C CTRL-B sequence.

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

This pseudo-macro has not yet been 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

SEARCH LIST TOO LONG

You've specified a search list which is too long. The search list cannot exceed 511 characters.

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 VALUE FORMAT ERROR

You've entered an improperly formatted switch.

SWITCH UNKNOWN

You entered an invalid switch.

TERMINATED BY ERROR

This process was terminated by an error.

TOO MANY CHARACTERS IN STRING

The STRING argument exceeds 127 characters.

TOO MANY REMOTE FILES OPEN

UNABLE TO ACCESS QOUTPUT FILE

UNABLE TO ACCESS QLIST 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.

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 <.

UTILITY TASK STACK OVERFLOW

This is a system error; see your system manager.

WRONG NUMBER OF ARGUMENTS

You've supplied the wrong number of arguments to a command or pseudo-macro.

ZERO DIVISOR

You've supplied a zero divisor to the [!UDIVIDE] pseudo-macro.

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. This packet is described in Table B-1.

Table B-1. ?GTMES Parameter Packet

Offset	Contents
?GREQ	Table B-2 of your programmer's manual shows request types
?GNUM	Argument number. Argument number zero indicates the program name, when CLI format is returned
?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. The returned string is always terminated by a null

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 left slash and must end with a null 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.

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

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
- characters which delimit literals are removed
- if the command was XEQ, EXECUTE, DEBUG, PROCESS, or CHAIN, the command is removed

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.

?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 zero. 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.

```
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.
```

Figure B-1. Sample Error Return Routine

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.

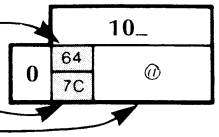
End of Appendix

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 16	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.

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 *AOS Operator's Guide* or the *AOS/VS Operator's Guide.*

Table D-1. Optional Batch Job Switches

Switches	Effect
/CPU=hh*[:mm[:ss]]*	Limit the amount of CPU time this batch job can use. Hours (hh) are required; 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-78 for May 6, 1978). You must specify all three parts of the date. Time is in the form hh*[:mm[:ss]]*. Hours (hh) are required; 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, 1978, use the following format: $$JOB/AFTER=3-8-78:14 username
/AFTER=+time	Delay processing until a specific amount of time has elapsed. Time is in the form hh*[:mm[:ss]]*. You can omit the seconds or the minutes and seconds. For example, to begin processing a job in one week, use the following format: $$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.

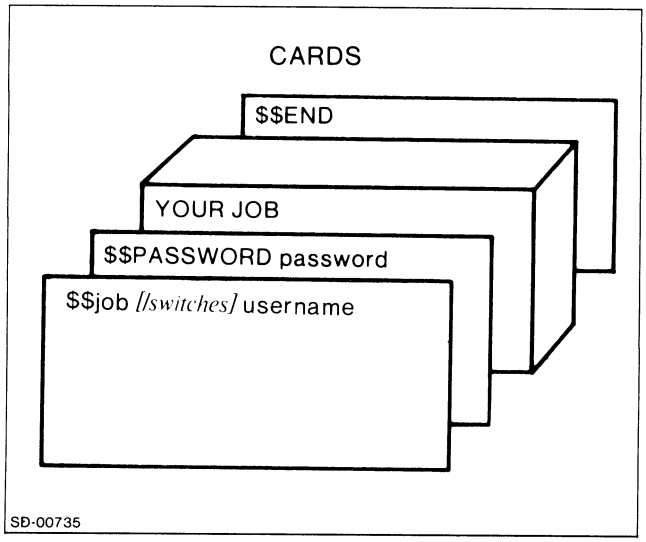


Figure D-1. Stacked Format for Batch Jobs

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 is completed, the system deletes the file. The system will also delete the file if the batch job is unsuccessful.

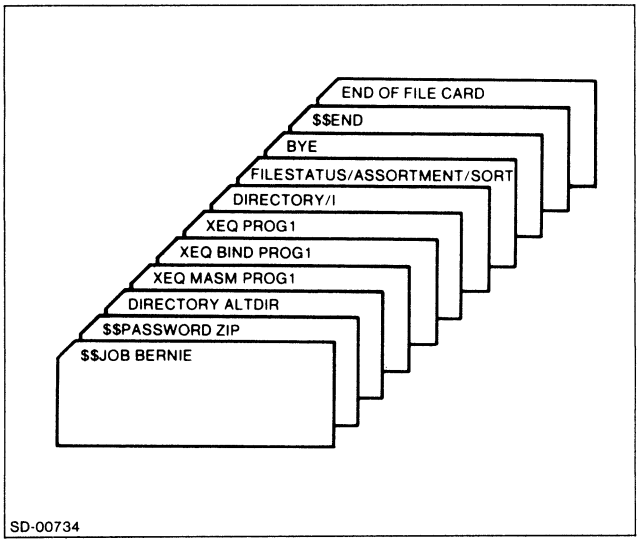


Figure D-2. Sample Batch Job in Stacked Format

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 *AOS Operator's Guide,* (093-000194) or the *AOS/VS Operator's Guide* (093-000244) 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* (093-000260).

First Step -- Start System Logging

Issue the CLI SYSLOG command from the *operator's CLI, PID 2,* to start system logging. Note that you can *only* run SYSLOG from PID 2. For example, the command

```
) SYSLOG/START :NOV11.LOG↵
```

starts logging for the file NOV11.LOG. If you omit the filename, the system writes all logging information to the file :SYSLOG.

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, and /FA.

In addition, the /X switch lets you run all XODIAC™ logs on your report.

XODIAC Related Switches for REPORT

/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

/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

Index

Note: Underscored page numbers (e.g., 1-5) indicate definitions of terms or other key information.

D

E

F

M

N

O

P

Q

R

S

T

U

V

W

X

DataGeneral

users group

Installation Membership Form

Name _____ Position _____ Date _____

Company, Organization or School _____

Address _____ City _____ State _____ Zip _____

Telephone: Area Code _____ No. _____ Ext. _____

1. Account Category

- ☐ OEM
- ☐ End User
- ☐ System House
- ☐ Government
- ☐ Educational

2. Hardware

	Qty. Installed	Qty. On Order
M/600		
COMMERCIAL ECLIPSE		
SCIENTIFIC ECLIPSE		
AP/130		
CS Series		
Mapped NOVA		
Unmapped NOVA		
microNOVA		
Other (Specify) _____		
_____		

3. Software

- ☐ AOS
- ☐ DOS
- ☐ MP/OS
- ☐ RDOS
- ☐ Other

Specify _____

4. Languages

- ☐ Algol
- ☐ DG/L
- ☐ Cobol
- ☐ PASCAL
- ☐ Business BASIC
- ☐ BASIC
- ☐ Assembler
- ☐ Fortran
- ☐ RPG II
- ☐ PL/1
- ☐ Other

Specify _____

5. Mode of Operation

- ☐ Batch (Central)
- ☐ Batch (Via RJE)
- ☐ On-Line Interactive

6. Communications

- ☐ HASP
- ☐ RJE80
- ☐ RCX 70
- ☐ CAM
- ☐ XODIAC
- ☐ Other

Specify _____

7. Application Description

○ _____

8. Purchase

From whom was your machine(s) purchased?

- ☐ **Data General Corp.**
- ☐ Other

Specify _____

9. Users Group

Are you interested in joining a special interest or regional Data General Users Group?

○ _____

CUT ALONG DOTTED LINE

DataGeneral

Data General Corporation, Westboro, Massachusetts 01580. (617) 366-8911

FOLD

TAPE

FOLD

TAPE

FOLD

FOLD

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:

Data General

ATTN: Users Group Coordinator (C-228)
4400 Computer Drive
Westboro, MA 01581

DataGeneral

users group

Installation Membership Form

Name _____ Position _____ Date _____

Company, Organization or School _____

Address _____ City _____ State _____ Zip _____

Telephone: Area Code _____ No. _____ Ext. _____

1. Account Category

- ☐ OEM
- ☐ End User
- ☐ System House
- ☐ Government
- ☐ Educational

2. Hardware

	Qty. Installed	Qty. On Order
M/600		
COMMERCIAL ECLIPSE		
SCIENTIFIC ECLIPSE		
AP/130		
CS Series		
Mapped NOVA		
Unmapped NOVA		
microNOVA		
Other (Specify) _____		
_____		

3. Software

- ☐ AOS
- ☐ DOS
- ☐ MP/OS
- ☐ RDOS
- ☐ Other

Specify _____

4. Languages

- ☐ Algol
- ☐ DG/L
- ☐ Cobol
- ☐ PASCAL
- ☐ Business BASIC
- ☐ BASIC
- ☐ Assembler
- ☐ Fortran
- ☐ RPG II
- ☐ PL/1
- ☐ Other

Specify _____

5. Mode of Operation

- ☐ Batch (Central)
- ☐ Batch (Via RJE)
- ☐ On-Line Interactive

6. Communications

- ☐ HASP
- ☐ RJE80
- ☐ RCX 70
- ☐ CAM
- ☐ XODIAC
- ☐ Other

Specify _____

7. Application Description

○ _____

8. Purchase

From whom was your machine(s) purchased?

- ☐ **Data General Corp.**
- ☐ Other

Specify _____

9. Users Group

Are you interested in joining a special interest or regional Data General Users Group?

○ _____

CUT ALONG DOTTED LINE

DataGeneral

Data General Corporation, Westboro, Massachusetts 01580. (617) 366-8911

FOLD

TAPE

FOLD

TAPE

FOLD

FOLD

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:

Data General

ATTN: Users Group Coordinator (C-228)
4400 Computer Drive
Westboro, MA 01581

ISD User Documentation Remarks 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 *Command Line Interpreter (CLI) User's Manual (AOS and AOS/VS)* Manual No. *093-000122-05*

Who are you?

- ☐ EDP Manager
- ☐ Senior Systems Analyst
- ☐ Other _____
- ☐ Analyst/Programmer
- ☐ Operator

What programming language(s) do you use? _____

How do you use this manual? *(List in order: 1 = Primary Use)*

- ____ Introduction to the product
- ____ Reference
- ____ Other _____
- _____ Tutorial Text
- _____ Operating Guide

About the manual:	Yes	Somewhat	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 have any comments on the software itself, please contact your Data General Systems Engineer. If you wish to order manuals, see your Data General Sales Representative.

Remarks:

Date _____

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

ISD User Documentation, M.S. E-111
4400 Computer Drive
Westborough, Massachusetts 01581

ISD User Documentation Remarks 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 *Command Line Interpreter (CLI) User's Manual (AOS and AOS/VS)* Manual No. *093-000122-05*

Who are you?

- ☐ EDP Manager
- ☐ Analyst/Programmer
- ☐ Senior Systems Analyst
- ☐ Operator
- ☐ Other _____

What programming language(s) do you use? _____

How do you use this manual? *(List in order: 1 = Primary Use)*

- ____ Introduction to the product
- ____ Tutorial Text
- ____ Reference
- ____ Operating Guide
- ____ Other _____

About the manual:	Yes	Somewhat	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 have any comments on the software itself, please contact your Data General Systems Engineer.
If you wish to order manuals, see your Data General Sales Representative.

Remarks:

Date _____

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

ISD User Documentation, M.S. E-111
4400 Computer Drive
Westborough, Massachusetts 01581

Data General Corporation, Westboro, MA 01580