MP/AOS

Command Line Interpreter (CLI)

Data General

MP/AOS

Command Line Interpreter (CLI)

DataGeneral

Notice

Ordering No. 069-400201
© Data General Corporation, 1982
All Rights Reserved
Printed in the United States of America
Rev. 00, April 1982

Preface

This manual describes the user's main interface with the MP/AOS operating system. The text has been written to provide a clear explanation of CLI concepts and features. It is supplemented by detailed examples, tables, and illustrations. An extensive command dictionary provides the experienced programmer with a quick reference to command descriptions, formats, and examples.

Organization of this Manual

This manual is divided into nine chapters and three appendices as described below.

Chapter 1, "Introduction," describes the basic features of the CLI.

Chapter 2, "CLI Command Line Syntax," describes the different types of commands available in the CLI and explains their syntax.

Chapter 3, "The File System," details the tree-structured file organization under MP/AOS, and explains the use of the CLI for file management.

Chapter 4, "CLI Environment," delineates the different parameters forming the CLI environment, and explains the CLI commands allowing user modification of these parameters and of some global system parameters.

Chapter 5, "Process Management," introduces MP/AOS multiprocessing concepts and explains how to create and manage processes using CLI commands.

Chapter 6, "Macros, Pseudomacros, and Textual Substitutions," discusses the creation of macros to execute CLI command sequences. Explains how to enhance the function of a CLI command or macro by using pseudomacros and/or textual substitutions.

Chapter 7, "Arithmetic and Conversion Functions," describes CLI arithmetic operations and conversion commands to ASCII, octal, and decimal.

Chapter 8, "CLI Command Summary," summarizes the CLI commands in tabular form, grouped by related concepts.

Chapter 9, "CLI Command and Pseudomacro Dictionary," describes the function and format of each CLI command and pseudomacro, and its corresponding global and command-specific switches. Examples are provided for all entries. The dictionary is organized in alphabetic order.

Appendix A, "CLI Error Codes," lists the error codes and corresponding text messages for all errors that occur in the CLI.

Appendix B, "The ASCII Character Set," lists the ASCII character set supported by MP/AOS and the corresponding octal, decimal, and hexidecimal equivalents.

Appendix C, "I/O Device Mnemonics," lists the I/O devices accessible via the CLI.

An Index follows the Appendices.

At the back of the book you can find a list of Data General offices worldwide; addresses and phone numbers of agencies from which manuals and order forms can be obtained, and a comment form. Finally, a User's Group Membership form invites you to join the User's Group, which brings DGC software users together, through group meetings and publications, to exchange ideas, applications, problems, and solutions.

Reading the Manual

If you are new to Data General software, we suggest you continue with the text portion of the manual (Chs. 2-8) which provides an overview of CLI concepts and facilities.

Readers familiar with the AOS or MP/OS CLI may want to take particular note of the following MP/AOS CLI features:

- Conversion functions (Chapter 7)
- Pseudomacros (Chapter 6)
- Textual substitution facility (Chapter 6)
- RENAME allows re-grafting of files and empty directories within the directory structure (Chapter 3)
- FILESTATUS has added the ability to display the set of files accessed before or after a certain time or period of time, and to sort the file display by other information than filenames (Chapter 3)
- CLI environments are supported under the MP/AOS CLI (Chapter 4)
- New process control commands (Chapter 5)

Once you are familiar with MP/AOS CLI concepts and commands, you may want to refer to various portions of the manual only as you need them. Chapter 9, "CLI Command and Pseudomacro Dictionary," offers a complete, easy-to-use reference to command descriptions, formats, and examples.

Conventions and Abbreviations

Throughout this manual we use the following conventions to illustrate instruction formats:

COMMAND	Uppercase letters in THIS typeface indicate an instruction mnemonic. You type an instruction mnemonic exactly as it appears.
argument	Lowercase italic letters represent a command's argument. You must replace this symbol with the exact code for the argument you need.
[optional]	Brackets denote an optional argument. (Command switches appear in this format as well.) If you use this argument or switch, do not type the brackets into your command line: they only set off the choice.
CTRL-	Depress and hold the Control key while you press the character following CTRL-.
arg1| *arg2*	Denotes either *arg1* or *arg2*.
EXAMPLE LINE	Uppercase letters in THIS TYPEFACE are used for programming examples.
RESPONSE	If the program can respond to the command in the example the response is shown in uppercase letters in *THIS TYPEFACE*.

Related Manuals

The following manuals also belong to the series of books published on the MP/AOS operating system.

MP/AOS Concepts and Facilities (DGC No. 069-400200) provides a concise but thorough introduction to the MP/AOS operating system for users who want to assess the system's advantages.

MP/AOS System Programmer's Reference (DGC No. 093-400051) documents MP/AOS system software and provides a complete dictionary of system calls and library routines.

Loading MP/AOS (DGC No. 069-400207) describes how to install MP/AOS software on ECLIPSE-line computers.

MP/AOS System Generation and Related Utilities (DGC No. 069-400206) describes the generation of an MP/AOS system tailored to specific applications. It also describes the following utilities, including sample dialogues as appropriate:

- SYSGEN, the interactive system generation utility;
- DINIT, the disk initializer;
- FIXUP, the disk repair utility;
- SPOOLER, which controls line printer operations;
- ELOG (error logger), the utility for interpreting the system log file.

MP/AOS Debugger and Performance Monitoring Utilities (DGC No. 069-400205) describes the following utilities, providing a dictionary of debugger commands and sample dialogues as appropriate:

- FLIT, the process debugger;
- PROFILE, which measures execution-time performance;
- OPM, the process monitor that displays current system resource allocation and status.

MP/AOS Macroassembler, Binder, and Library Utilities (DC No. 069-400210) documents that MP/AOS macroassembler and binder as well as the library file editor (LED) and system cross-reference analyzer (SCAN). It includes programming examples and a dictionary of assembler pseudo-ops.

MP/AOS Advanced Program Development Utilities (DGC 069-400208) describes the text control system (TCS), which can maintain multiple versions of a file, select the correct version to build a program, and find text files.

MP/AOS SPEED Text Editor (DGC No. 069-400202) documents the features of SPEED, the MP/AOS character-oriented text editor.

MP/AOS SLATE Text Editor (DGC No. 069-400209) documents the features of SLATE, a screen- and line-oriented text editor.

MP/AOS File Utilities (DGC No. 069-400204) describes the following utility programs, providing sample dialogues for each:

- FEDIT, a file editor that permits modification of system files, including program and data files;
- FDISP, which can display the address and data contents of a file or compare two files, displaying the parts that differ;
- SCMP, which can compare two source programs line by line;
- MOVE, which allows the transfer of files among directories;
- AOSMIC, which allows manipulation of MP/AOS and MP/OS disks and files on an AOS system;
- FOXFIRE, which permits the transfer of files among MP/OS, MP/AOS, and AOS systems over asynchronous communications lines.

SP/Pascal Programmer's Reference (DGC No. 069-400203) documents an extended Pascal for system programmers. SP/Pascal has all of the features of MP/Pascal as well as special features targeted for the MP/AOS and AOS operating systems.

Books on three additional programming languages supported by MP/AOS have previously been published as part of the bookset for the MP/OS operating system:

MP/Pascal Programmer's Reference (DGC No. 069-400031) documents for system programmers a Pascal-based language, targeted for the MP/OS operating system.

MP/FORTRAN IV Programmer's Reference (DGC No. 069-400033) documents for system programmers a language based on ANSI 1966 standard FORTRAN, with extensions.

MP/BASIC Programmer's Reference (DGC No. 069-400032) documents for new users a programming language based on ANSI standard BASIC with extensions.

Contents

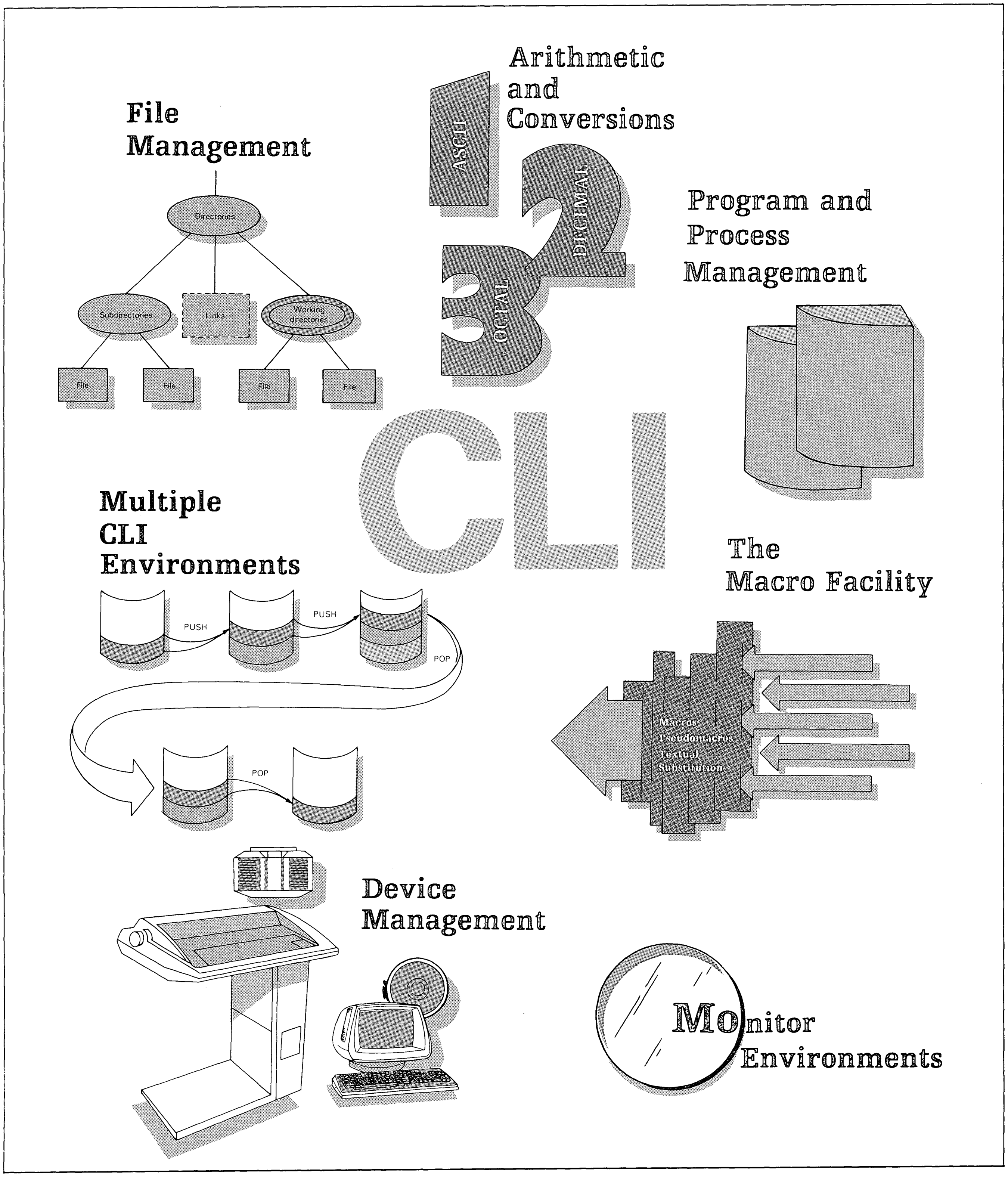


Figure 1.1 The CLI: interface to operating system and program development tool

What is the CLI?

The Command Line Interpreter (CLI) is your primary interface with the MP/AOS operating system. It is an interactive program that enables you to manage the MP/AOS file system, control various aspects of the CLI environment and system configuration, invoke other utilities and user programs, manage processes and programs, and perform arithmetic and conversion operations.

The CLI macro facility enables you to create and execute command sequences. You can enhance commands and macros by using pseudomacros and textual substitutions. On-line information about CLI commands and topics can be displayed at your terminal via the HELP facility.

Managing MP/AOS Files

The MP/AOS file system consists of a collection of peripheral devices, directories, and data files organized in a hierarchical structure. The CLI enables you to define the system's organization, create new files, and reference or manipulate any file in the system.

Invoking Utilities & High-level Languages

The CLI enables you to invoke the utilities and high-level languages listed in Table 1.1, as well.

Type	Utilities and High-level Languages
Program Development Utilities	Binder (BIND)
	Command Line Interpreter (CLI)
	File Constructer (BUILD)
	Histogrammer (PROFILE)
	Library Editor (LED)
	Macroassembler (MASM)
	Pattern Search Program (FIND)
	Process Debugger (FLIT)
	Process Monitor (OPM)
	System Cross-Reference Analyzer (SCAN)
	Slate Text Editor (SLATE)
	Text Control System (TCS)
	Speed Text Editor (SPEED)
File Management Utilities	File Display (FDISP)
	File Editor (FEDIT)
	File Transfer Utility (FOXFIRE)
	Move Utility (MOVE)
	Source Compare Program (SCMP)
System Management Utilities	Disk Initializer (DINIT)
	Error Logger (ELOG)
	Disk Repair (FIXUP)
	Line Printer Spooler (SPOOLER)
	System Generation (SYSGEN)
High-Level Languages	MP/Basic
	MP/Fortran IV
	MP/Pascal
	SP/Pascal

Table 1.1 MP/AOS utilities and high-level languages accessible via the CLI

For more information about these utilities, refer to the appropriate manual. There is a list in the *Related Manuals* section in the Preface.

Managing CLI Environments

The CLI environment affects how you and the CLI interact. You can control certain aspects of this environment, such as: the branches of the file system automatically searched when you attempt to access a file, the CLI response to command errors, the characteristics of your console terminal, and the environment level at which you are processing.

You can also modify a few of the operating system parameters. The system date and time can be changed, and disk devices can be mounted or dismounted. These parameters can be changed without affecting other activity in the system. From the CLI in the initial (first created) process, you can also bootstrap another operating system.

Multiprocessing

The MP/AOS operating system supports multiprocessing: you can create, debug, stop, restart, or terminate processes from the CLI. CLI commands also enable you to display status information and send messages from one process to another.

Executing Programs

Each process in the MP/AOS operating system executes a program (such as the CLI). You can execute other programs from the CLI by temporarily writing the CLI out to disk, or by overwriting the CLI with the new program.

Arithmetic and Conversion Operations

You can perform basic arithmetic operations in the CLI: add, subtract, multiply, divide, and modulus. Also available are commands to convert from octal to decimal, decimal to octal, and octal to ASCII. These operations are most useful within macros.

Using the Macro Facility

The macro facility enables you to write a sequence of CLI commands into a file to be executed whenever you invoke the macro filename as a CLI command. A macro is particularly useful for frequently-executed command sequences. You can increase the flexibility of a macro by including in it pseudomacros and textual substitution.

Textual Substitution

Output from a CLI command or macro can stand in as the argument to another command or macro: this facility is called textual substitution. It allows you to perform several operations, each modifying the next, in one command line.

Using the HELP Facility

The HELP command supplies either brief information about the syntax and use of a command you specify, or a longer explanation of that command or a topic.

To review a list of the CLI topics and commands on which HELP can provide information, type HELP without arguments.

```
) HELP ↓
```

The CLI responds with a list of HELP topics and commands.

HELP followed by a command name displays brief information on a command.

```
)HELP command ⤶
```

The following command line supplies the full name of the command, and a list of all the switches available for that command.

```
)HELP QPR ⤶
QPRINT    -ARGUMENT(S) REQUIRED
   SWITCHES /1= /2 = /L(=) /BANNER= /COPIES= /FOLDLINES /QUEUE=
   /NOTIFY
FOR MORE HELP TYPE
HELP/V QPRINT
)
```

To request a detailed explanation of a command, use the /V switch and the name of the command.

```
)HELP/V command ⤶
```

To get more information from HELP, the Help file must be available for the item, and the help directory must be on the searchlist. If the Help file is not available, the CLI displays:

```
No more help available.
```

HELP followed by an asterisk and a topic name (from the list displayed by HELP without arguments) displays information on the specified topic. (Don't forget the asterisk!)

```
)HELP *topic ⤶
```

If the CLI has no information for the item you specified, it displays the message:

```
item    -Unknown
```

Getting Started in the CLI

To use the CLI, you must first turn on the power to the computer and bootstrap the MP/AOS operating system. Refer to *Loading and Generating MP/AOS* (DGC No. 069-400207) for bootstrapping instructions.

Once MP/AOS has been loaded, it will automatically execute the CLI program and display two messages at the console:

```
MP/AOS    REV. 1.00
MP/AOS    CLI REV. 1.00 SWAPLEVEL 1
)
```

The first message lets you know that the MP/AOS operating system is running. The other indicates that the CLI is executing. After these messages are displayed, a right parenthesis is displayed to prompt you for a CLI command.

NOTE: *The revision numbers appearing in messages displayed at your console may differ from those shown here.*

You can edit command lines and modify your current interaction with the CLI by using control characters and control sequences.

Control Characters

Control characters are transmitted to the CLI when you press and hold the CTRL key and at the same time press another key on the keyboard. A control character is represented in this manual as CTRL-*X*, where *X* is the other key you press with CTRL. Control characters are usually echoed (that is, displayed) as ↑*X*. MP/AOS control characters are listed in Table 1.2.

Control Characters	Function
CTRL-C	Starts a control sequence. See Table 1.3.
CTRL-D	Indicates the end of a file. Terminates file input from the console.
CTRL-O	Discards output to the terminal. Toggles; that is, a subsequent CTRL-O restores terminal output.
CTRL-Q	Cancels the effect of CTRL-S by restarting output to the terminal.
CTRL-S	Suspends output to the terminal ("freezes" the screen display).
CTRL-T	Retypes the current line.
CTRL-U	Deletes the current command line.

Table 1.2 Control characters

CTRL-S stops the display of information on the terminal and suspends the task when it attempts to write to the terminal.

If you type a CTRL-S by accident, you may think that the system is dead because nothing you enter affects the console. When this happens, type CTRL-Q. If CTRL-S caused the problem, CTRL-Q will negate its effect.

CTRL-O discards information written to the terminal. If a task is writing data to the terminal when you type this control character, the task 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 you type CTRL-O again, when a task reads from or performs a forced write to the terminal, or when the process terminates.

CTRL-T is useful when you have typed several deletes at a hard-copy terminal. This control character will re-display the command line without the deleted characters.

CTRL-U cancels the current input line (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 CLI echoes ↑U.

Control Sequences

A control sequence is a CTRL-C followed by another control character. The control sequences available in MP/AOS are listed in Table 1.3.

Control Sequence	Function
CTRL-C CTRL-A	Signals a console interrupt that is program-dependent
CTRL-C CTRL-B	Terminates the program currently executing
CTRL-C CTRL-E	Terminates the current program and creates a break file containing the memory image of the terminated program

Table 1.3 Control sequences

If you type CTRL-C CTRL-A when the executing program is the CLI, the CLI will display an interrupt error message followed by the prompt,), on the next line.

Typing CTRL-C CTRL-B terminates the currently executing program unless you are processing the CLI at Program Level 1 in the initial process. In this case, CTRL-C CTRL-B shuts down the system. (Processes and program levels are explained in Chapter 5.)

The CTRL-C CTRL-E sequence is primarily useful when you are debugging programs.

Monitoring CLI Activity

Certain CLI commands help keep track of your activity in the CLI. Since each command you enter generates a five-digit error code (successfully executed commands return a zero), you can display the error code for the previously entered command using the ERCODE command. To display the text message that corresponds to the error code, enter the MESSAGE command. Refer to Appendix A for a listing of MP/AOS error codes and their corresponding text messages.

Because the CLI, as any other program executing in the MP/AOS system, executes at some program execution level, you can issue the SWAPLEVEL command to display the level at which the current CLI is processing. Refer to Chapter 5 for more about this command and program execution levels.

Each command you enter in the CLI takes a certain amount of time to execute. You can measure this time by turning on the Elapsed Time Reporting mode with the MEASURE command.

To debug your macros, you can turn on Trace mode with the TRACE command; each command line will then be displayed prior to its execution by the CLI. This command and its use in macros are discussed in Chapter 6.

For a detailed description of each of these commands and examples of their use, refer to the command descriptions in the "CLI Command Dictionary," Chapter 9.

Pausing the CLI

If you need to delay the CLI temporarily, use the PAUSE command. This command stops further processing in the CLI for the time period you specify.

Exiting from the CLI

You can terminate the CLI by entering the BYE command.

```
) BYE↵
```

which displays

```
MP/AOS CLI TERMINATING
```

NOTE: *The message displayed at your terminal may differ slightly from the example, depending on your version of the CLI.*

The commands you can use to perform CLI functions are explained as you proceed through the text. CLI syntax conventions are presented in the next chapter.

CLI Command Line Syntax

2

This chapter explains how to enter commands according to CLI syntax conventions. Full discussions of each command appear in "CLI Command Dictionary," Chapter 9.

To execute a command, you enter the command name and follow it with appropriate switches and arguments. The command name can be abbreviated, but the abbreviation must be unique.

Switches may be included with a command to select specific options or to modify the normal command function. The CLI standard switches can be included with any CLI command. Some commands accept additional switches specific to those commands.

The arguments you include with a command specify values to be passed to the command when it executes. Some commands require arguments, other commands accept arguments, and still others do not use arguments.

The CLI attempts to interpret each command line it encounters. A command line can contain one or more commands, and can be any length, extending to continuation lines if necessary.

Parentheses and angle brackets can be used as short-cuts to repeat a command for different arguments, group arguments in an argument list, or expand arguments containing the same characters.

The CLI provides you with a set of commands to perform many of the basic operations available in MP/AOS. The function of these commands can be enhanced by using additional facilities: pseudomacros and textual substitution. These features are discussed in Chapter 6.

Command Evaluation

You can enter CLI commands when the CLI prompts for input with). The CLI checks the first entry in the command line to see if it corresponds to:

an explicit CLI macro call,
a CLI command,
a CLI pseudomacro or textual substitution,
a CLI macro with or without the .CLI extension,
a program name with the .PR extension, or,
a program name without the .PR extension.

If the first entry in the command line is equivalent to any of these, the command, macro, or program is executed. If, however, all attempts to match the entry fail, the CLI displays

```
Unknown command, macro or program name
```

Abbreviating Commands

You can abbreviate CLI commands and switches if you wish. The shortest abbreviation allowed consists of the fewest characters (beginning with the first character) that uniquely identify the command or switch. (This is called the minimal unique abbreviation.)

For example, you can abbreviate the FILESTATUS command to F because it is the only CLI command beginning with an F. You cannot, however, abbreviate DELETE to D, because several commands begin with D. Even DE would be insufficient because of the command DEBUG. The shortest acceptable abbreviation for DELETE is DEL.

NOTE: *It is advisable to spell out command names that you incorporate into macros. This eliminates the need for changing the macro when new commands are implemented that might render your abbreviations obsolete.*

Switches

A switch is a predefined character or character sequence that alters the function of a command or argument. You can append one or more switches to a CLI command or argument to select different processing options or to 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.

Simple Switches

A simple switch has this format:

/switchname

For example, the normal function of the COPY command is to copy the contents of one file to a new file. The /A switch modifies the command to append the contents of one file to the contents of an existing file.

```
) COPY/A TRIG COSINE ↵
```

appends the contents of file COSINE to those of file TRIG.

Keyword Switches

A keyword switch has the format

/keyword=value

For example, /TYPE=*n* is a keyword switch for the CREATE command. You enter a value following the equal sign.

```
) CREATE/TYPE=DIR TERMINALS ↵
```

creates a directory file named TERMINALS.

If you specify a keyword switch that requires a date, time, or time-period value, format its value according to Table 2.1.

Value	Format*
date	*dd-mon-yy*
time	*:hh:mm:ss*
time-period	*-dd[:hh:mm]*

Table 2.1 Date/time format for keyword switches

** In these formats:* dd *is days,* mon *is month,* yy *is year,* hh *is hours,* mm *is minutes, and* ss *is seconds. The square brackets enclose optional values.*

For example, the /AFTER/TLM=*date[time]* keyword switch that is used with the FILESTATUS command requires a value for the date and (optionally) the time. To specify January 1, 1981 as the date and noon as the time, enter

```
) FILESTATUS/AFTER/TLM=1-JAN-81:12:00:00
```

The /AFTER/TLM=*time-period* switch accepts a time period. To specify a time period of two days, enter

```
) FILESTATUS/AFTER/TLM=-2
```

CLI Standard Switches

There are five switches that can be used with any CLI command. These "standard" switches are listed in Table 2.2.

Switch	Action
/1=*error-response*	Sets the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Sets the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Writes CLI output to the line printer (@LPT) instead of to the default output device.
/L=*pathname*	Writes CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Reports the time used to execute this command (to the nearest second).

Table 2.2 CLI standard switches

The /1=*error-response* and /2=*error-response* switches override the CLI's normal error processing, but only for the duration of the command you enter. Ordinarily, the CLI handles errors according to the CLASS1 and CLASS2 values established for the CLI environment, as discussed in Chapter 4.

Both /L and /L=*pathname* direct the CLI to send command output somewhere other than the console: to the printer (@LPT) or to a file, respectively. (Note that /L does not use the spooler for printing.) For example,

```
) TYPE/L REPORT
```

prints the contents of file REPORT on the line printer instead of displaying the contents at the console.

The /TIMEMODE switch requests the CLI to display how long it takes to execute the command. For instance, to display the time required to execute the DATE command (which displays the system date), enter

```
) DATE/TIMEMODE
12-MAY-81
01
```

The DATE command executed in 1 second.

In addition to the standard switches, some commands accept other switches that apply only to those commands. Each command description in the "CLI Command Dictionary", Chapter 9, includes a list of command-specific switches.

Command Arguments

An argument is a character string included with a command. In general, an argument tells the command what data to process. Different commands expect different types of arguments. For example, one command may expect a file name, while another might require an integer. Some commands require one or more arguments, some commands accept arguments but do not require them, while still other commands do not accept arguments.

Argument Delimiters

If the command line you enter contains one or more arguments, you must use delimiting characters to separate the arguments from the command and from each other. Table 2.3 lists the CLI delimiters.

Delimiter
One or more spaces
One or more tabs
A single comma
Any combination of the above (except for multiple commas)

Table 2.3 CLI delimiters

Suppose you want to enter the CREATE command with ROBOTICS as an argument. If you separate the command from the argument with a CLI delimiter, the command line could look like any of the following:

```
) CREATE ROBOTICS ↵

) CREATE,ROBOTICS ↵

) CREATE      ROBOTICS ↵
```

Literal Strings

If you include a text string enclosed in double quotation marks (") as an argument to a command, the CLI treats everything between the quotes as a single argument. It performs *no* interpretation of punctuation, and does no special character processing (such as angle brackets expansion, or textual substitutions).

For example,

```
) XEQ PILOT.PR ''VALUES <P,Q,R>''
```

Command Terminators

A command is not transmitted to the CLI for execution until you terminate it properly. Use one of the terminators listed below to end a command line.

New-line
Form Feed
Carriage Return
CTRL-D

This book represents the New-line terminator as a curved arrow, ⤶.

Multiple Commands

You can type more than one CLI command on a single line by separating the commands with a semi-colon (;). For example,

```
) DIRECTORY FUNDS;DELETE ACCT ⤶
```

Commands are executed from left to right, as if they were on separate command lines. Hence, in this example the DIRECTORY command is executed, and then the DELETE command.

If a command causes an error, the remainder of the command line may or may not be processed, depending upon the current error response values set for CLI exception conditions (refer to "Handling CLI Exception Conditions" in Chapter 4).

Continuing Command Lines

If you are entering a lengthy command line, you can continue it to the next line by typing an ampersand (&) immediately before typing the New-line on the current line. In other words, the ampersand is the last character on the line before you go to the continuation line. The CLI responds by issuing the prompt &) on the continuation line.

NOTE: *The ampersand is not a delimiter. If you need a delimiter at the point where you continue the line, include a delimiter before the ampersand or at the beginning of the continuation line.*

For example,

```
) DELETE MULTIPLEXORS SYNCHRONOUS & ⤶
&)ASYNCHRONOUS COMMUNICATIONS ⤶
```

This command line continues to a second line. The DELETE command deletes the four files listed. The space before the & delimits SYNCHRONOUS and ASYNCHRONOUS as two separate arguments.

There is no limit to the number of continuation lines. However, a single CLI command line can contain no more than 2048 characters.

Repetition, Grouping, & Expansion

Using parentheses and angle brackets, you can repeat a command for each of the arguments you include, group arguments together, and/or expand arguments.

Repeating a Command

If you type 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 with the command on a separate command line. You can follow or precede a parenthetical expression with any character. For example, the command line:

```
) WRITE (X Y Z) ↓
```

is equivalent to

```
) WRITE X ↓
) WRITE Y ↓
) WRITE Z ↓
```

NOTE: *The WRITE command is useful to experiment with arguments in parentheses (and with angle brackets in the next section). WRITE displays the arguments you enter as they are interpreted by the CLI.*

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 ↓
```

is equivalent to

```
) WRITE A,C ↓
) WRITE B,C ↓
```

If you enter a command with two or more argument groups in parentheses, the CLI executes the command on the first argument in each group, then on the second argument in each group, and so on. For example,

```
) WRITE (A,Z),(B,Y) ↓
```

is equivalent to

```
) WRITE A,B ↓
) WRITE Z,Y ↓
```

When a group is exhausted, the CLI executes the command on the remaining arguments in other groups (if any remain). Command repetition ends when the CLI executes the last argument in the largest argument group. For example,

```
) WRITE (A,B,C),(X,Y) ↵
```

is equivalent to

```
) WRITE A,X ↵
) WRITE B,Y ↵
) WRITE C ↵
```

If you enter a command line with two or more commands in parentheses, followed by an argument, the CLI executes each command in the list with the argument.

```
) (TYPE DELETE), FILEQ ↵
```

is equivalent to

```
) TYPE FILEQ ↵
) DELETE FILEQ ↵
```

Grouping Arguments

Nesting one set of parentheses within another isolates the nested set from the outermost set of arguments. That is, the CLI treats the nested arguments as a group. For example,

```
) WRITE (A,(B,C),D) ↵
```

is interpreted as

```
) WRITE A ↵
) WRITE B,C ↵
) WRITE D ↵
```

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

Expanding Arguments

Angle brackets help you code arguments that contain the same character or character combination. The CLI forms arguments by joining each character enclosed within angle brackets with the characters that appear immediately before the left angle bracket and immediately after the right angle bracket. For example,

```
) DELETE FILE<P,Q,R> ↵
```

is equivalent to

```
) DELETE FILEP FILEQ FILER ↵
```

Consider the next example that contains two bracketed groups.

```
) WRITE  <1,2><1,2> ↵
```

is equivalent to

```
) WRITE 11,12,21,22 ↵
```

In this example, the first argument is combined with each argument in the second bracketed group, and then the second argument in the first group is combined with each argument in the second.

You can also nest angle brackets. The CLI processes the angle brackets from left to right, and from inner to outer bracket pairs. The following example shows one pair of brackets nested within another pair:

```
) WRITE  1,2<3,4<5,6>> ↵
```

The CLI first processes the inner pair:

```
) WRITE  1,2<3,45,46> ↵
```

Next the outer pair is processed, thereby displaying

```
1 23 245 246
```

If arguments 1 and 2 are also placed in angle brackets:

```
) WRITE  <1,2><3,4 <5,6>> ↵
```

the CLI processes the command as:

```
) WRITE  <1,2><3,45,46> ↵
13 145 146 23 245 246
```

Using Parentheses With Brackets

You can include parenthetical groups and bracketed groups in the same command line. You can also nest angle brackets within angle brackets, parentheses within parentheses, parentheses within angle brackets, and angle brackets within parentheses. The CLI follows a specific order to evaluate parenthetical and bracketed groups.

The CLI first processes angle brackets from left to right and from inner to outer. When no angle brackets are left in the command line, the CLI processes parentheses in pairs from left to right. Remember, when the command line contains embedded parentheses, the outermost parentheses are interpreted as a repetition operator, the second-level as a grouping operator, third level as a repetition operator, etc. Consider the following examples.

Example 1

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

First, the CLI expands this command line to:

```
) WRITE (11,21) (12,22) ↵
```

Then, the CLI repeats the WRITE command for the two argument groups, as in:

```
) WRITE 11 12 ↵
) WRITE 21 22 ↵
```

Example 2

```
) 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:

```
) WRITE (11,12,21,22) ↵
```

The CLI then repeats the WRITE command for each argument in the parentheses. The original command line in Example 2 would actually be interpreted as:

```
) WRITE 11 ↵
) WRITE 12 ↵
) WRITE 21 ↵
) WRITE 22 ↵
```

Example 3

Suppose you embed parentheses within angle brackets as in this example:

```
) WRITE <A,(B,C)>D ↵
AD BD
AD CD
```

The CLI processes the angle brackets from left to right, expanding this command to:

```
) WRITE AD,(B C)D ↵
```

Then the CLI interprets the parentheses pairs, from left to right, thereby resulting in:

```
) WRITE AD,BD ↵
) WRITE AD,CD ↵
```

Example 4

Example 4 is more complex, nesting parentheses within parentheses and angle brackets within parentheses as well.

```
) WRITE (FILE<1,2,3>,((B,C)A)) ↵
FILE1
FILE2
FILE3
BA
CA
```

First the CLI processes the angle brackets.

```
) WRITE (FILE1,FILE2,FILE3,((B,C)A)) ↵
```

Then the parentheses are processed.

```
) WRITE FILE1 ↵
) WRITE FILE2 ↵
) WRITE FILE3 ↵
) WRITE (B,C)A ↵
```

The last parentheses are then expanded to

```
) WRITE BA ↵
) WRITE CA ↵
```

As well as the syntax conventions discussed thus far, additional rules must be followed when entering pseudomacros, textual substitutions, and macros. The remainder of this chapter briefly describes these syntax rules.

Macros

A macro is a file that contains a sequence of CLI commands, pseudomacros, or textual substitutions. Each time you invoke a macro, the CLI executes the command sequence contained in the macro file. Although you use a macro as if it were a CLI command, there are additional syntax conventions you must follow to invoke it. Chapter 6 covers macros in detail.

To execute the macro's command sequence, enter the macro name as the first entry in the command line, using one of the following formats:

- **Explicit invocation:** enclose the macro name and any arguments in square brackets. If the macro name contains the .CLI extension, enter the name with or without the extension.
- **Implicit invocation:** enter the macro name and any arguments WITHOUT square brackets. Again, if the name of the macro contains the .CLI extension, enter the name with or without the extension.

For example, to invoke macro LOCATE.CLI with argument QUAD4, you can type any of the following:

```
) [LOCATE.CLI QUAD4] ⤶

) [LOCATE QUAD4] ⤶

) LOCATE.CLI QUAD4 ⤶

) LOCATE QUAD4 ⤶
```

You can use explicit invocation to avoid conflict between a macro file and a CLI command of the same name, e.g.,

```
[DATE.CLI] ⤶
```

calls a macro, not the CLI DATE command.

A macro can also be written to function as an include file.

Include Files

An include file is a macro containing a sequence of file names, rather than a sequence of commands. (The sequence of file names in an include file *must* end with a CTRL-D character, *not* a New-line.) This type of macro is always invoked as an argument to a command. You invoke the macro by entering its name in square brackets. For example, if MINI.CLI contains the names of three files,

```
STRUCTURE,IMPLEMENT,APPLICATION
```

and you want to display the contents of all three files with the TYPE command, enter

```
) TYPE [MINI.CLI] ⤶
```

or

```
) TYPE [MINI] ⤶
```

Pseudomacros

The pseudomacro facility provides you with functions that add further dimension and control to macro command sequences. Some pseudomacros execute single instructions, while others process a sequence of instructions when a given condition is satisfied. This facility is explained in Chapter 6.

A pseudomacro cannot function alone as a command in the CLI. It is used either within the context of a macro or as an argument to a CLI command.

The first character in each pseudomacro name is an exclamation point (!). You enter a pseudomacro by enclosing it in square brackets [ ]. For example, !EQUAL, !ELSE, and !END are pseudomacros that appear in the following example:

```
[!EQUAL %1% %2%]
  WRITE THE ARGUMENTS ARE THE SAME
[!ELSE]
  WRITE THE ARGUMENTS ARE NOT THE SAME
[!END]
```

Any arguments to the pseudomacro must appear within the brackets, e.g., %1% and %2% are arguments to the !EQUAL pseudomacro in this last example.

Textual Substitution

Textual substitution redirects the output from one CLI command to function as an argument to another command. Textual substitutions are always entered as arguments to other CLI commands. This mode is also discussed in Chapter 6.

Although pseudomacros and textual substitutions are different in function, the syntax for both is similar. Enter a textual substitution as:

[!*command*]

where "command" is any CLI command or macro name. For instance,

```
WRITE [!STRING] ↲
```

The STRING command functions as a textual substitution to the WRITE command. The CLI executes the STRING command first, and uses the resulting output as an argument string to the WRITE command.

The File System

The MP/AOS file system is a hierarchical structure of directories and subordinate files. CLI commands enable you to access and manipulate files in the system. You can create different types of files, copy one file to another, rename, or delete a file. The contents of a file can be displayed at your terminal, and you can list various statistics about a file.

In this chapter, all the concepts of file identification are introduced *before* most of the file management commands are discussed. Note that you can access a file only by preceding the file specification with a suitable command or macro.

This chapter discusses file system concepts as they apply to MP/AOS and also explains how you can manage your files. The CLI commands you can use to manage your files are listed in Table 3.1.

Command	Action
ATTRIBUTES	Set or display a file's attributes
COPY	Copy one or more files into a destination file
CREATE	Create a file
DELETE	Delete one or more files
DIRECTORY	Display or modify the working directory.
FILESTATUS	Display file status information
PATHNAME	Display the fully-qualified pathname of a file
QPRINT	Place one or more files in the spooler output queue
RENAME	Change the name of a file, or graft files or empty directories onto another part of directory tree
SEARCHLIST	Set or display the searchlist
TYPE	Type the contents of one or more files

Table 3.1 File management commands

File System Characteristics

This section explains file-naming conventions, the file hierarchy organization, and file types.

Filenames

Each file is identified by a filename. Filenames may consist of up to 15 of the following characters:

a through z
A through Z
0 through 9
? $. _

The system does not distinguish between upper- and lower-case alphabetic characters.

Certain conventions are used to interpret filenames you enter, or to name the files created on your behalf. Certain filename extensions (suffixes) represent specific file types. These are listed in Table 3.2 below. You can invent other extensions to suit your needs.

Filename Extension	Interpretation
.BAS	MP/BASIC source file
.CLI	CLI macro file
.FR	MP/FORTRAN IV source file
.OB	Object file
.OL	Overlay file
.PAS	MP/Pascal and SP/Pascal source file
.PR	Executable program file
.PS	Permanent symbol table file
.SR	Assembly language source file
.ST	Symbol table file

Table 3.2 Interpretation of filename extensions

Examples of valid MP/AOS file names:

- Z
- CITY
- Transport.pr
- MODELS.LIST
- FRIDAY_STATUS
- ?007$93_AB

Directories

Files are grouped into directories. A directory is a file that can contain names of other files and directories (subdirectories). Each directory functions as a reference point for locating other files in the system.

You can nest directory files to any level. A directory containing subdirectories is "superior" to those subdirectories, and is therefore considered to be their "parent" directory in the *file hierarchy,* or *tree-structured* file system.

Figure 3.1 illustrates a sample file system.

NOTE: *The level of directory nesting is limited only by the 127-character limit for pathnames.*

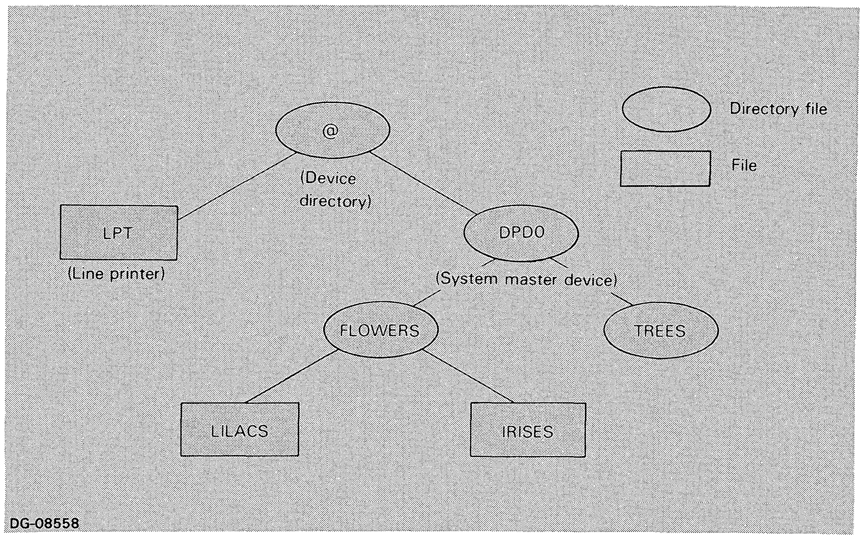


Figure 3.1 Hierarchical file system

In this figure, FLOWERS and TREES are subdirectories of directory DPD0: .

A directory has information about only the files and subdirectories it contains. It does not have information about files contained in other directories in the system.

Device Directory

The highest level in the MP/AOS hierarchical file system is the device directory, which is named @. This directory is a table in the system's memory containing the filenames of all the input/output devices in the system. This directory is *not* accessible via the DIRECTORY command.

The name the system associates with a device is the device's filename preceded by @. The file system in Figure 3.1 contains two devices: a line printer, identified by @LPT, and the system master device, @DPD0. The system master device is the disk from which you initially load (bootstrap) the MP/AOS operating system.

Non-disk devices, such as @LPT, do not contain directories. Disk devices, though, such as @DPD0, can contain directories and files organized in a tree structure.

Root Directory

This tree structure originates in the disk's *root directory.* The root directory is the highest directory in the hierarchical directory structure on the disk, and is identified by the disk name followed by a colon, e.g., @DPD0:. The colon differentiates the root directory from the device, itself. Hence, you use @DPD0: to refer to the root directory on device @DPD0.

System Master Device

In every MP/AOS system, one disk device is designated as the system master device. This is the device from which the system is booted.

If you have only one disk device in your system, this disk will be the system master device by default, as is the case with device @DPD0 in Figure 3.1.

Figure 3.2 shows two disk devices, @DPD0 and @DPD1, both containing user files and directories. @DPD0 is designated the system master device.

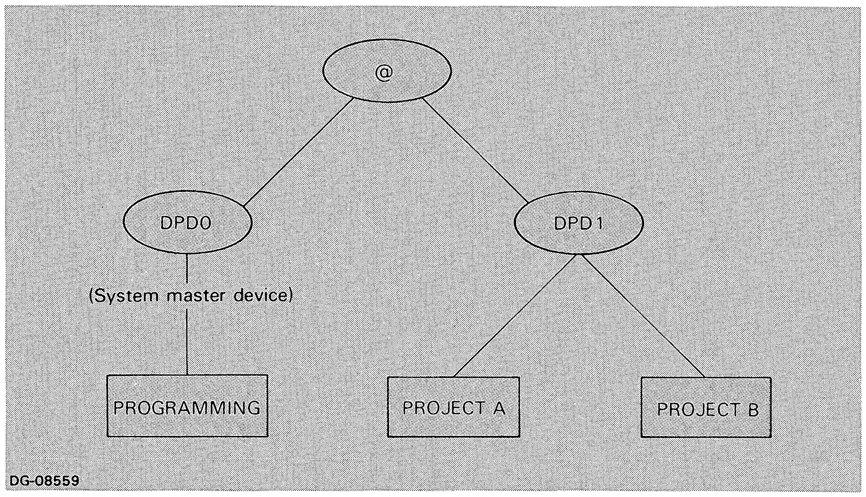


Figure 3.2 A file system with two disk devices

You have the option of referring to the system master device with a colon (:) instead of its full name, *@devicename:* .

The examples in this chapter assume that the files are stored on the system master device.

File Types

Besides directory files, MP/AOS supports several other file types, as listed in Table 3.3. Each file type is identified by a three-character mnemonic and a file-type number.

Identifying Mnemonic	File Type
BPG	MP/OS format bootable program*
BRK	Program break file
DIR	Directory
IDF	MP/ISAM file
IXF	MP/ISAM index file
LIB	Library
LNK	Link file
LOG	System log file**
MBS	MP/BASIC save file
OBF	Object file
OLF	Overlay file
PRG	Program file
PST	Permanent symbol table (used by assembler)
STF	Symbol table file
TXT	Text file
UDF	User data file

Table 3.3 MP/AOS file types and mnemonics accessible via the CLI

**Currently not bootable under MP/AOS.*

***LOG can be shown via FILESTATUS but not created.*

Many of these file types may be created with the CREATE command, discussed later in this chapter. However, BPG and BRK files must be created at the system level. See the *MP/AOS System Programmer's Reference* (DGC No. 093-400051).

File Attributes

A file can be assigned different attributes to stipulate how the file can be used. File attributes available in MP/AOS are listed in Table 3.4.

Attribute	Use of File
A	The file is attribute-protected; its attributes cannot be changed.
P	The file is permanent; it cannot be renamed or deleted.
R	The file is read-protected; it may not be read.
W	The file is write-protected; it may not be written to.

Table 3.4 MP/AOS file attributes

You cannot assign attribute A through the CLI. It is set exclusively by the system. This attribute applies only to devices and root directories on disk devices.

You can, however, set and modify the R, W, and P attributes using the ATTRIBUTES command. The format for this command is:

)ATTRIBUTES *filename [attribute-list]* ⤸

Replace *attribute-list* with R, W, and/or P. If you do not include an attribute list, the CLI displays the current attributes for the file.

The system initially sets the permanence attribute (P) for the root directory and all other directories when they are created.

The following example checks the attribute setting of a file named FISCAL, then sets its attributes to W to prevent FISCAL from being modified.

```
) ATTRIBUTES FISCAL ⤸
FISCAL NO ATTRIBUTE PROTECTION
) ATTRIBUTES FISCAL W ⤸
) ATTRIBUTES FISCAL ⤸
FISCAL W
```

You can delete all attribute protection by including the /K switch with the ATTRIBUTES command.

File Space Allocation

MP/AOS allows you to optimize file storage on disk devices by controlling the size of file elements. A file element is either a single disk block (512 bytes) or a group of physically contiguous disk blocks. The system allocates and deallocates file space in elements rather than blocks.

You can specify a file's element size when you create it. A large element size causes the data in the file to be organized in large units of space; hence, reading and writing the file can be done more efficiently. On the other hand, a small element size gives the system greater flexibility in allocating disk blocks. Choose an element size to compromise between speed and efficient use of space.

If you do not specify an element size when you create a file, the CLI assigns an element size of one disk block.

Identifying Files

The CLI enables you to identify files using complete and abbreviated pathnames in conjunction with the CLI working directory, searchlist, link files, and templates.

Pathnames

A *pathname* represents the unique route through the file system to a specific file. If you are referencing a file on the system master device, the complete or *fully-qualified* pathname begins with the system master device name (e.g., @DPD0:), followed by one or more filenames separated by colons. All the filenames except the last must be the names of directories, and each directory named except the first must be a subdirectory of the preceding directory.

In Figure 3.1, the pathname to IRISES is

@DPD0:FLOWERS:IRISES

You can abbreviate the name of the root directory *on the system master device* to a colon (:). You can thus enter the pathname for IRISES as:

:FLOWERS:IRISES

If you are identifying a file on a device other than the system master device, you must begin the pathname with the full name of the device starting with the @ sign. Therefore, in Figure 3.2, the pathname to PROJECT.A is

@DPD1:PROJECT.A

Pathnames can contain as many as 127 characters and therefore can become cumbersome. Sometimes it is more convenient to reference your files according to their relationship to the working directory, or through the use of the searchlist.

The Working Directory

The working directory is a movable reference point within the MP/AOS file structure. At any particular time, the directory you've specified as your working directory is the one in which, or from which, your work in the CLI is accomplished. Any directory in the file structure accessible to you can become your working directory.

Whenever you specify a file name instead of a pathname in the command line, files you create, delete, or rename are in the current working directory. When you retrieve a file, the working directory is always the first location searched, unless you specify a pathname in your command line. Finally, the working directory is an important reference point when retrieving files from elsewhere in the file structure.

The directory you are "in" when you first start executing the CLI is called the *initial* working directory. When you first bring up the system, the initial working directory is : . (For any CLI other than the initial process, the initial working directory is the working directory of the process that created it, at the time of creation. Processes are described in more detail in Chapter 5.)

You use the DIRECTORY command to verify and set your working directory. (For more on setting the working directory, refer to *Using the Working Directory,* later in this chapter.)

In Figure 3.3, the working directory is shown as BROOKLYN. To verify this, you would type:

```
) DIRECTORY ⤶
BROOKLYN
```

The CLI displays the name of the working directory.

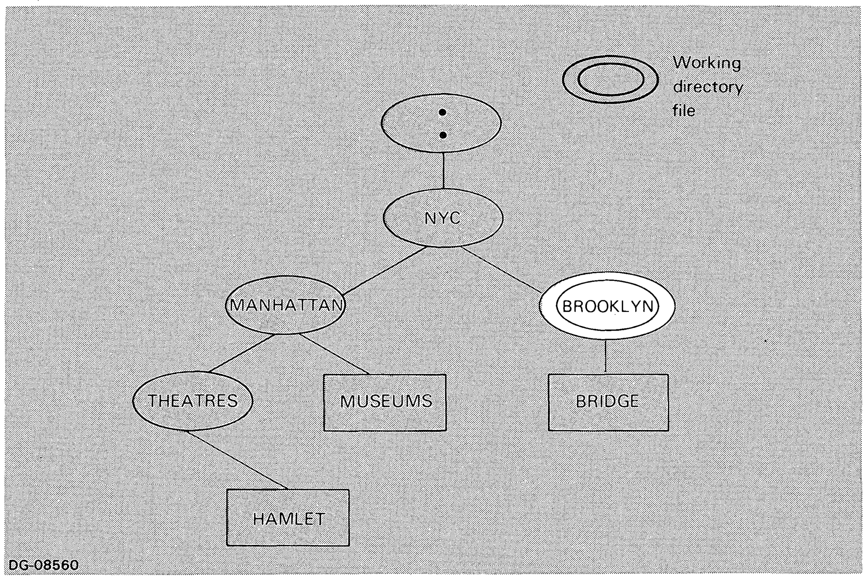


Figure 3.3 Sample directory tree

To display the file BRIDGE with the TYPE command, you can enter the complete pathname, as in

```
)TYPE :NYC:BROOKLYN:BRIDGE ⤶
```

However, BRIDGE is in the working directory, so you can simply enter:

```
)TYPE BRIDGE ⤶
```

To specify a file in the parent directory of the working directory, precede the file name with a ↑ (circumflex or up-arrow, depending on your terminal).

Suppose that the working directory in Figure 3.3 is THEATERS. To reference HAMLET, you enter its filename. However, if you want to access file MUSEUMS, which is in the parent directory, MANHATTAN, you must refer to it by entering either its complete pathname

:NYC:MANHATTAN:MUSEUMS

or the abbreviated pathname

↑MUSEUMS

If, instead, you want to access directory BROOKLYN, you will have to enter two up-arrows, as in

↑↑BROOKLYN

Partial Pathnames

If you want to indicate a file in a subtree of the working directory, you can use a partial pathname. A partial pathname begins with a subdirectory of the working directory and is followed by one or more filenames, ending with the file you are indicating. Separate the names with colons.

Refer to Figure 3.3. Suppose the working directory is NYC. To access HAMLET, you can enter the partial pathname

MANHATTAN:THEATERS:HAMLET

To access BRIDGE in directory BROOKLYN, you can enter the partial pathname

BROOKLYN:BRIDGE

rather than its complete pathname

:NYC:BROOKLYN:BRIDGE

Pathname Prefixes

Four characters can be used as prefixes to pathnames. These are : (colon), = (equal sign), ↑ (up-arrow), and @ (at sign). Each prefix is interpreted according to Table 3.5.

Pathname Prefix	System Interpretation
: (colon)	Start at the root directory of the system disk. This makes the pathname start at the apex of the directory tree.
= (equal sign)	Limit the system to searching for the file in the working directory.
↑ (up-arrow)	Move up to the parent directory.
@ (at sign)	Start at the device directory.

Table 3.5 Pathname prefixes

To use these prefixes, make sure the pathname specifies all the directories between the one specified by the prefix and the one you want to reference.

For instance, in Figure 3.3, assume the working directory is BROOKLYN. To refer to the directory MANHATTAN using the prefixes listed in Table 3.5, any of the following pathnames is correct:

@DPD0:NYC:MANHATTAN
(starts at the root directory of the disk @DPD0)
:NYC:MANHATTAN
(starts at the root directory of the system master device)
↑MANHATTAN
(starts at the parent of the working directory)

From the working directory BROOKLYN, you can refer to the file BRIDGE in any of the following ways:

@DPD0:NYC:BROOKLYN:BRIDGE
(starts at the root directory of the disk @DPD0)
=BRIDGE
(restricts the search to the working directory)
BRIDGE
(looks in the working directory first, and finds it there)

The Searchlist

The searchlist specifies a list of directories in a certain order that the system can check if a file (specified only by its simple filename) does not exist in the working directory. When you identify a file by its simple filename, the system searches for it in your working directory. But if the file is not there, the system looks for it in the directories listed in the searchlist. If the same simple filename exists in more than one of the directories in the searchlist, the system stops searching as soon as it finds that filename in a directory in the searchlist. You can modify the searchlist to search directories containing frequently used files.

Initially, the searchlist contains the root directory of the disk from which the system is booted. To display the current searchlist, enter SEARCHLIST. For example,

```
) SEARCHLIST ↲
:NYC:BROOKLYN,:NYC:MANHATTAN
```

indicates that the searchlist currently contains the name of two directories in Figure 3.3.

Suppose you are in directory THEATERS but want to display the BRIDGE file contained in directory BROOKLYN using the TYPE command.

```
) TYPE BRIDGE ↲
```

The system will first look in the working directory for BRIDGE. Because this directory does not contain the file, the system then looks through each directory listed in the searchlist (in the order they were entered). The system will find the file in directory :NYC:BROOKLYN.

If you wish to restrict a file search to your working directory and not have the searchlist scanned, include the = prefix before the filename.

NOTE: *The searchlist is not searched when you:*

- *enter a fully-qualified pathname*
- *include a pathname prefix with the filename (such as = or ↑; see Table 3.5)*
- *enter the CREATE, DELETE, RENAME, or FILESTATUS command; in this case, the search is limited to the working directory.*

As long as the file you are accessing has a unique filename and is in one of the directories in the searchlist or in the current working directory, you can retrieve the file by filename, instead of by pathname.

To modify the searchlist, refer to *Using the Searchlist.*

Displaying a Complete Pathname

To display the complete pathname of a file, use the PATHNAME command. This command searches first the working directory and then the directories in the searchlist for the specified file. The format for this command is

PATHNAME *partial-pathname* ⤶

If the file cannot be found in the working directory or searchlist directories, the CLI responds with an error message. Consider Figure 3.4.

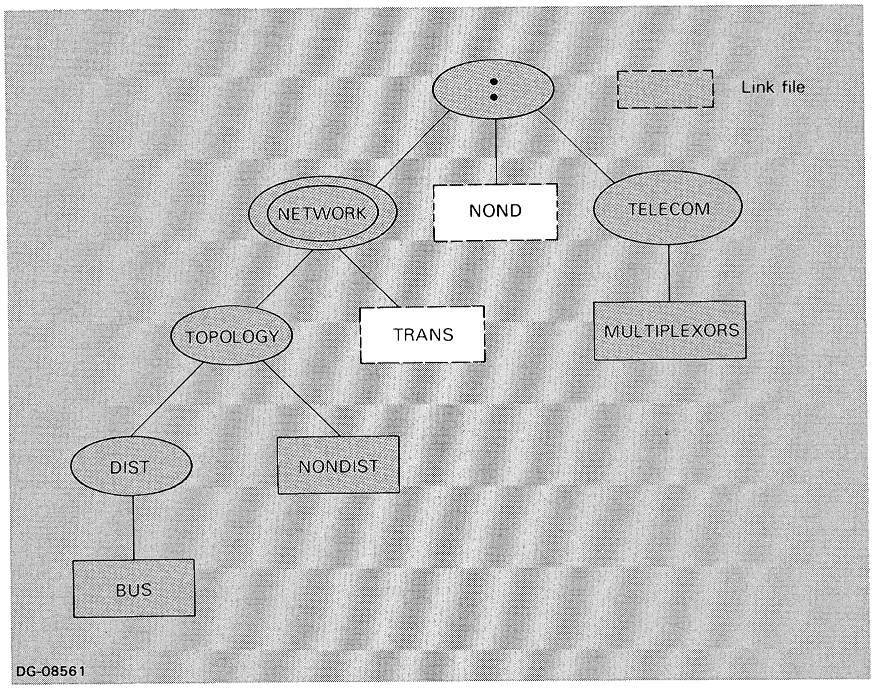


Figure 3.4 Directory tree containing link files

```
) DIRECTORY ⤶
@DPDO:NETWORK
) PATHNAME MULTIPLEXORS ⤶
@DPDO:TELECOM:MULTIPLEXORS
```

This example displays your working directory and then the complete pathname for file MULTIPLEXORS. Notice the file is not in the working directory. Therefore, the CLI found the file by using the searchlist.

Using Links

Another facility that simplifies file access is the link. A link is a file used to represent a pathname or pathname segment. When you include the link name in a pathname, the system resolves the linkname to the contents of the link file.

For example, suppose in Figure 3.4 that NETWORK is the working directory, and TRANS is a link file containing pathname segment TOPOLOGY:DIST. If you enter pathname TRANS:BUS, the system will interpret this as

TOPOLOGY:DIST:BUS

Now suppose you have changed the working directory to TELECOM. To retrieve file BUS from here, either of the following pathnames is correct:

```
:NETWORK:TOPOLOGY:DIST:BUS    (uses a fully-qualified pathname)
↑NETWORK:TRANS:BUS     (uses a pathname prefix and the link)
```

A link can resolve to any file (if the link's content is resolvable). In Figure 3.4, NOND is a link that represents the pathname segment NETWORK:TOPOLOGY:NONDIST. The following pathname starts at the root directory, and so can be used to select the file NONDIST *no matter which* directory is the working directory.

:NOND

A link file can be created with the CREATE command. Use this format:

CREATE/LINK *linkname link-contents* ⤶

The first argument specifies the name of the link file, and the second indicates the pathname you want the file to contain.

Note that the CLI does not check the syntax of the link contents when you create it. You can create a link that does not contain a resolvable or valid text stream.

NOTE: *The FILESTATUS command does not resolve a link to its contents.*

Using Templates

A filename template is zero or more legal filename characters plus one or more of the CLI's special template characters. Templates may be used as shortcut arguments with certain CLI commands.

Template characters function as "wild cards." That is, the CLI interprets each character in the filename template literally except for the template character. The template character tells the CLI to access all files in a particular directory that match the template's criteria. Table 3.6 lists and explains the template characters.

Template Character	Represents
* (asterisk)	Any single character except for a period or null string
- (hyphen)	Any character string (including the null string) that does not contain a period
+ (plus sign)	Every character string, including those containing periods and the null string
\ (back slash) used with another template or filename	An exclusion of files matching the template or filename following the backslash

Table 3.6 Filename templates

Look at Figure 3.5 for the following examples. Assume that the working directory is MAP.

Suppose you want to display the contents of files in the working directory whose filenames are similar. If you enter a filename without a template character, only that file will be displayed:

```
) TYPE FILEA ↲
```

displays FILEA only.

However, if you enter

```
) TYPE FILE* ↲
```

the CLI looks for every file whose filename consists of FILE plus one more character, excluding those containing a period or null string. FILEA and FILEB are the only files matching this template; hence these files would be displayed.

Entering the command as

```
) TYPE FILE- ↲
```

will display any file whose filename is FILE plus any character string lacking a period. This command line displays FILEA, FILEAB, FILEB, and FILEXY.

To display the contents of all files whose names begin with FILE, enter the following command:

```
) TYPE FILE+ ↲
```

This displays FILEA, FILEAB, FILEB, FILEXY, FILE.1, and FILE.2.

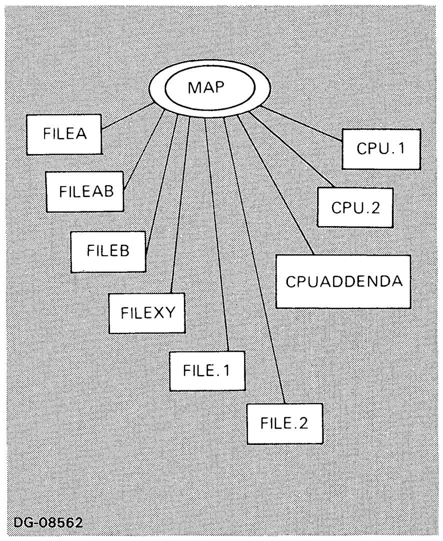


Figure 3.5 Template example

To display all the files whose names contain an A, enter

```
)TYPE +A+ ⤶
```

This displays FILEA, FILEAB, and CPUADDENDA.

To exclude a file from those that match the specified template, include a backslash with the file or template you do not want.

```
) TYPE FILE+\FILEA ⤶
```

displays all the files displayed by TYPE FILE+ except for FILEA.

If you enter

```
) TYPE CPU.- ⤶
```

the CLI displays all files whose names are CPU. followed by a character string containing no periods: CPU.1 and CPU.2. The template character specifies the criteria for only the remaining characters in the filename.

Entering

```
) TYPE CPU+ ⤶
```

displays the contents of CPU.1, CPU.2, and CPUADDENDA.

If you want to type all the files in the directory, enter the plus sign only.

```
) TYPE + ⤶
```

displays all the files in directory MAP.

Manipulating Files

Thus far, this chapter has described the MP/AOS file system and how you can access files in it. The remainder of this chapter will show you how to create files; manipulate the working directory and searchlist; copy, rename, or delete a file; display the contents of a file; and display information about a file.

Creating Files

The structure of your file system is determined by the files you create. As well as directory files, you can also create other file types. Those types accessible from the CLI are listed in Table 3.3. You can use either the CLI CREATE command or a text editor, such as SPEED, to create a file. (SPEED can create *only* text files.) The format for the CREATE command is

)CREATE *pathname* ⤶

Including the /DIR switch creates a directory file. For example,

```
) CREATE/DIR RESOURCES ⤶
```

creates a new directory named RESOURCES, subordinate to the working directory. To create a file of another type, include the /TYPE=*n* switch, substituting a mnemonic from Table 3.3.

If you do not use switches with this command, the system creates an empty text file (TXT) with an element size of 1 block (512 bytes). To specify an element size other than the default, include the /ELEMENTSIZE=*n* switch.

NOTE: *The system automatically turns on the permanence (P) attribute when you create a directory file. This means directories cannot be renamed or deleted until you turn off this attribute. Refer to "File Attributes," earlier in this chapter.*

Entering Text in a File

You can enter text into a new file when you create it by including the /I switch with CREATE. This switch tells the CLI that you will enter input into the new file from the keyboard. For example, create a text file named REVOLUTION ready to receive text as follows:

```
)CREATE/I REVOLUTION ↵
))
```

The system will respond with a *))*, for keyboard input.

Enter the text you want, ending each line with a New-line. Before you type the New-line, correct any typing mistakes you have entered. Once you have terminated a line, you cannot return to that line while using CREATE/I.

To indicate there is no more input, enter a single right parenthesis as the first and only character on the line, and terminate the line. When a line begins with a right parenthesis, the CLI strips it from the line; if it was the only character on the line other than the line terminator, input from the keyboard is ended. If the) was not the only character on the line, the CLI continues to accept input for the file, and will prompt for further input on the next line. If you want a) as the first character in a line, enter it as)).

```
))THIRTEEN AMERICAN COLONIES ↵
))REBELLED AGAINST KING GEORGE ↵
))) ↵
```

To create a file that does not end in a New-line, terminate the last line with CTRL-D (end of file).

NOTE: *If you enter CREATE with the /M switch within the body of a macro file (macros are explained in Chapter 6), the file will accept input from subsequent lines of the macro. The last line to input must contain only a single).*

CREATE/I can be used to enter text only in a new file. To modify an existing file, use a text editor, such as SPEED.

Using the Working Directory

As already described, the working directory is your current directory location in the file system. It functions as a reference point. The system locates files and other directories according to their position relative to the working directory.

You can change the working directory to any directory on any disk currently mounted in the system. To change or display the working directory, use the DIRECTORY command.

)DIRECTORY *[pathname]* ↲

Entering DIRECTORY by itself displays the pathname of the working directory.

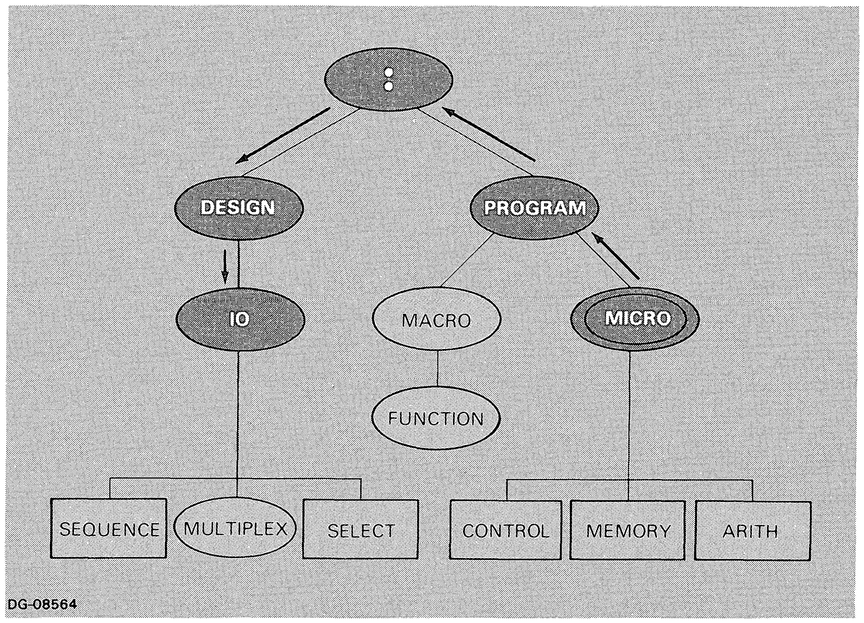


Figure 3.6 Path to another directory

```
) DIRECTORY ↲
:PROGRAM:MICRO
```

If you enter

```
) DIRECTORY :DESIGN:IO ↲
```

the working directory will be changed to IO.

You can change the working directory to a superior directory with the ↑ symbol for each directory you ascend through. For example, if MICRO is again the working directory in Figure 3.6, entering

```
) DIRECTORY ↑↑ ↲
```

moves the working directory up two directories to the root directory, : . If instead, you want to change the working directory from MICRO to MACRO, you could enter

```
) DIRECTORY ↑MACRO ↵
```

To change the working directory to a subordinate directory, such as from MACRO to FUNCTION, just specify the path between them as a partial pathname.

```
) DIRECTORY FUNCTION ↵
```

Returning to Previous Working Directories

In any CLI session, the /I and /P switches for the DIRECTORY command provide shortcuts to former working directories.

If you enter DIRECTORY/I without an argument, the CLI changes the working directory to the initial working directory. Suppose that :PROGRAM is the initial working directory in Figure 3.6. Entering the following series of DIRECTORY commands

```
) DIRECTORY ↵
:DESIGN:IO
) DIRECTORY/I ↵
:PROGRAM
```

displays the current working directory, changes it to the initial working directory, and then verifies this change.

Entering DIRECTORY/I with an argument tells the CLI to change the working directory to the specified subdirectory of the initial working directory. If there is no such subdirectory, the working directory remains unchanged and the error is reported. For example, to change the working directory to MICRO, a subdirectory of the initial working directory (:PROGRAM) in Figure 3.6, enter

```
) DIRECTORY/I MICRO ↵
```

Using the Searchlist

The initial searchlist for the initial process contains the root directory of the disk from which the system is loaded. For any other CLI, the initial searchlist is passed to the CLI from the process that created it. (Processes are described in more detail in Chapter 5.) To display or change the searchlist, use the SEARCHLIST command.

)SEARCHLIST *[pathname-1 [... pathname-5]]* ↵

Entering SEARCHLIST without arguments displays the current list of directories in the searchlist.

```
) SEARCHLIST ↵
@DPDO:
```

The searchlist can contain up to five directories. The directories are searched according to the order you enter them. If a directory in the searchlist contains subdirectories that you want searched, the subdirectories as well must be included in the searchlist.

```
) SEARCHLIST :DESIGN ↵
```

changes the searchlist to :DESIGN (part of the directory tree shown in Figure 3.6.)

To append directories to the current searchlist, include the /A switch. The arguments you specify will be placed at the end of the current list. For example,

```
) SEARCHLIST/A :DESIGN:IO :PROGRAM:MACRO ↵
) SEARCHLIST ↵
@DPDO:DESIGN,@DPDO:DESIGN:IO,@DPDO:PROGRAM:MACRO
```

appends two directories, :DESIGN:IO and :PROGRAM:MACRO to the current searchlist, and displays the new searchlist. (Refer to Figure 3.6.)

To insert items in the middle of the searchlist, you re-specify the list. For example, to have :PROGRAM searched before :PROGRAM:MACRO in the searchlist in the example above, type:

```
)SEARCHLIST :DESIGN :DESIGN:IO :PROGRAM :PROGRAM:MACRO ↵
```

Note that if the directories to be searched contain a great many files, response time can increase for commands that use the searchlist.

To eliminate the current searchlist, enter SEARCHLIST/K without arguments.

The searchlist is one of the parameters defining the CLI user environment. The /P switch enables you to change the contents of the searchlist to those of the previous environment level. Exclude arguments when including this switch. Refer to Chapter 4 for more about the CLI environment.

Copying Files

The CLI COPY command enables you to copy one or more files (the source files) to another file (the destination file).

)COPY *destination-file source-file-1 [...source-file-n]* ↵

If the destination file does not already exist, the COPY command creates a file and copies the source file(s) into it. The time and date of last access and of last modification for the destination file are always set to the current system time.

The new file's specifications will depend on those of the first (or only) source file. When the source file is a disk file, the destination file will have the source file's specifications.

You can also copy from or to a peripheral device. If you copy from a peripheral device such as a terminal, to a new disk file, the new file will have the default specifications listed in Table 3.7.

File type	Text file
Element size	512 bytes
Time block	Time of last access and time of last modification are set to the current system time.

Table 3.7 Peripheral device to disk file default specifications

NOTE: *The COPY command does not apply to magnetic tape devices.*

If the destination file already exists and is not a peripheral device, you must include either the /A or /D switch with the COPY command. /A appends the contents of the source file(s) to the destination file. /D deletes the destination file, creates a new file with the same name and specifications as the old destination file, and copies the source file(s) into the new file. Including /A when copying to a character device suppresses the initial Form Feed, which would otherwise occur.

If you want to copy to or from a character device and the file you are copying contains control characters you don't want the system to interpret, include the /B switch. This turns on Binary mode for the duration of the COPY command, preventing control characters from being interpreted.

The following examples illustrate how you can copy files.

```
) COPY NUTRIENTS ORGANIC INORGANIC ⤶
```

Copies ORGANIC and INORGANIC into a new file, NUTRIENTS, that will have the same specifications as ORGANIC.

```
)COPY/A PROGRAM MODULE1 MODULE2 ⤶
```

Appends the contents of MODULE1 and MODULE2 to the contents of PROGRAM, which must already exist.

```
) COPY SECURITY @CON12 ⤶
```

Creates a disk file named SECURITY and copies the data entered at terminal @CON12 into it. To terminate the copying operation, the last line of data should end in a CTRL-D (end-of-file).

Renaming and Grafting Files

No two files in the same directory can have the same name, although files in different directories can.

To change the current name of a file, use the RENAME command.

)RENAME *current-pathname new-pathname* ↵

Current-pathname is the current name of the file, and *new-pathname* is the name to which you want the file changed. If *new-pathname* already exists, delete it by including the /D switch.

Suppose you want to copy a file called INSTRUCTIONS to another directory, but the directory already contains a file by the same name. You can rename one of the files so that both can reside in the same directory.

```
) RENAME INSTRUCTIONS RULES ↵
```

changes the name of file INSTRUCTIONS in the current working directory to RULES.

The RENAME command also lets you graft files and empty directories from one part of the directory tree to another. For example, suppose that you want to graft file UTRECHT in Figure 3.7 from the TREATIES directory to the FIELDING directory.

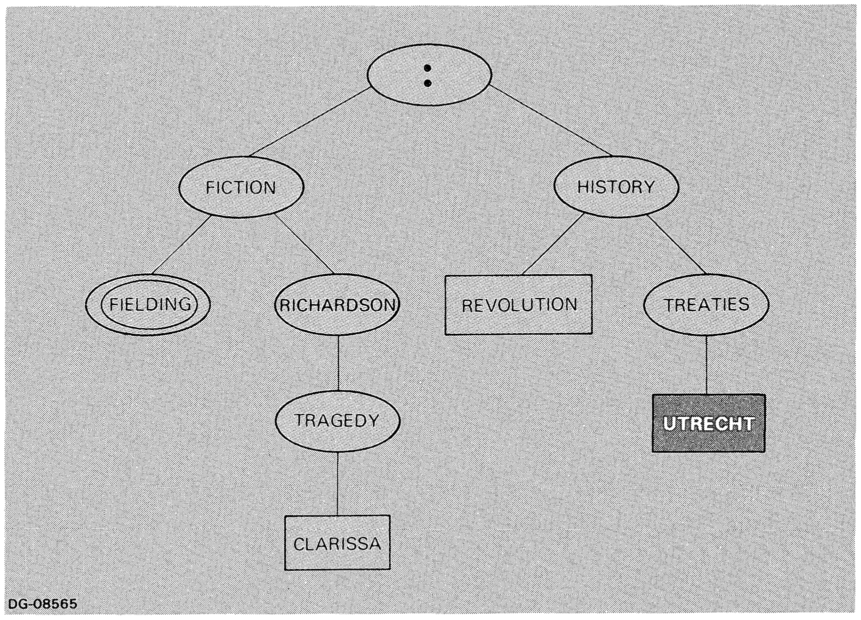


Figure 3.7 Directory tree before graft

Assuming that FIELDING is the current working directory, enter

```
) RENAME :HISTORY:TREATIES:UTRECHT UTRECHT ↵
```

The resulting directory tree is shown in Figure 3.8.

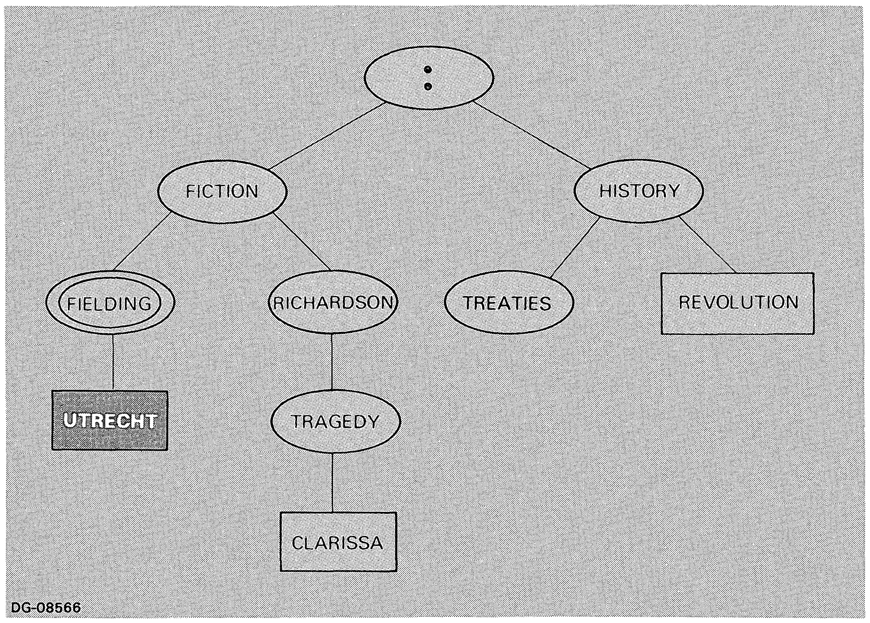


Figure 3.8 Directory tree after graft

Note the following renaming restrictions:

- The RENAME command treats partial pathnames as if they start in the working directory; the directories in the searchlist are not checked for the specified file.
- Only empty directories may be renamed.
- The permanence (P) attribute must be turned off before renaming a directory.
- Renaming across devices is not allowed.
- Renaming a link file does not affect the contents of a link (what it resolves to), only the name of the link file.

Deleting Files

You can use the DELETE command to delete a directory or a file.

)DELETE *pathname-1 [...pathname-n]* ↵

To delete an empty directory, turn off the P attribute first, using the ATTRIBUTES command, then delete the file.

The /DIR switch automatically turns off the P attribute for a directory file, and deletes the directory and any subtrees it contains.

To delete directory RICHARDSON and its subdirectories in Figure 3.8, enter

```
) DELETE/DIR RICHARDSON ↵
```

The DELETE command will accept templates in place of pathnames. This command never uses the searchlist to locate the files you specify.

To double-check a deletion, include the /C switch. This allows you to confirm the deletion by typing Y or to prevent the deletion by typing N or New-line.

To have the CLI verify each deletion, include the /V switch. For example:

```
) DELETE/C/V OPTICS ↵
OPTICS? Y ↵
DELETED OPTICS
```

This example asks if file OPTICS should be deleted. The CLI deletes OPTICS after receiving the confirming Y; then it verifies the action.

Displaying Files

To display the contents of a file at your console, use the TYPE command. This command does not allow you to modify the file's contents but only to examine them.

)TYPE *pathname-1 [...pathname-n]* ↵

TYPE accepts templates in place of pathnames. Using templates with simple filenames automatically confines the actions of TYPE to the working directory.

If you want the CLI to display the name of each file before typing it, include the /V switch.

```
)TYPE/V FILEX FILEY FILEZ ↵
```

will display the name and then the contents of each file in turn.

Printing Files

To print a file on the line printer, issue the QPRINT command.

)QPRINT *pathname-1 [... pathname-n]* ↵

The file will be placed in the lineprinter's queue. Be sure you do not delete or modify the file before it is output. To print more than one copy of each file, use the /COPIES=*n* switch, substituting the number of copies you need.

The arguments may be pathnames, filenames, or filename templates. You may specify one or more files to be printed.

If your system has more than one printer, you can specify the printer you want to use with the /QUEUE=*print_spooler* switch. If you don't use this switch, your file is printed on @LPT or @LPB, whichever is available.

The following command line produces two copies of file TITRATE.DATA on the device designated as LPT1.

```
) QPRINT/COPIES=2/QUEUE=LPT1 TITRATE.DATA ↵
```

A header page showing the file name, pathname, time last modified, and printing time will be printed at the beginning of the first (or only) copy of the file. With the /BANNER switch, you can add an optional text string up to 11 characters long to the information on the header page.

Normally, if a file contains lines that are longer than the paper width defined for the printer, the extra-long part of the line is simply truncated from the printed output. To have extra-long lines continued on the following lines, use the /FOLDLINES switch.

To receive a confirming message as soon as your file is printed, use the /NOTIFY switch.

For more information, see the *MP/AOS System Generation and Related Utilities Programmer's Reference* (DGC No. 069-400206).

Listing File Information

The FILESTATUS command lists information about files in the working directory.

)FILESTATUS *[pathname-1] [...pathname-n]* ↵

In response to this command, the CLI displays the pathname of the directory containing the files, followed by the name of each file specified and any file statistics requested in the switches.

If you omit arguments, the command will display information about all files in the working directory. If you exclude switches and arguments, the CLI displays only the names of the files.

You can enter filename templates in place of pathnames.

NOTE: *This command does not use the searchlist to locate file information.*

Two of the command switches are /ASSORTMENT and /SORT.

The /ASSORTMENT switch lists: file type, time, and date the file was last modified, and file length (in bytes). If you request information about a link file with the /ASSORTMENT switch, the CLI displays the linkname and its contents. FILESTATUS will not resolve a linkname to the filename it contains.

The /SORT switch displays the requested information in alphabetical order according to filename. If there is not enough space in memory to do a sort, the CLI will not attempt it; instead it will list the files in unsorted order.

Table 3.8 lists the various switches that you can include with FILESTATUS to display file information. FILESTATUS examples follow Tables 3.9 and 3.10.

Switch	Action
/AFTER	Used with the /TLA or /TLM conditional switch to display information collected after the specified time period.
/ASSORTMENT	Displays the file type, the time and date the file was last modified, and the file length in bytes.
/BEFORE	Used with the /TLA or /TLM conditional switch to display information collected before the specified time period.
/DLA	Displays the date the file was last accessed.
/DLM	Displays the date the file was last modified.
/ELEMENTSIZE	Displays the number of disk blocks in a file element for this file.
/LENGTH	Displays the file length in bytes.
/NHEADER	Does not display the pathname header(s) for the parent directories. Lists each file by pathname, not filename.
/SORT	Displays file names and associated information in alphabetical order, according to file name.
/SORT=field	Orders the file names to be displayed by one of the fields listed in Table 3.10.
/TLA	Displays the date and time the specified files were last accessed. A conditional form of this switch is described in Table 3.9.
/TLM	Displays the date and time the specified files were last modified. A conditional form of this switch is described in Table 3.9.
/TYPE	Displays the type of each requested file. A conditional form of this switch is described in Table 3.9.

Table 3.8 FILESTATUS command switches

The conditional switches in Table 3.9 display only those files that satisfy the stated condition.

Switches	Action
/AFTER/TLA= *[date][time] | time-period*	Display the file status of only those files last accessed on or after a specific date and time or during a given period of time.*
/BEFORE/TLA= *[date][time] | time-period*	Display the file status of only those files last accessed before a specific date and time or period of time.*
/AFTER/TLM= *[date][time] | time-period*	Display the file status of only those files last modified after a specific date and time or during a period of time.*
/BEFORE/TLM= *[date][time] | time-period*	Display the file status of only those files last modified before a specific date and time or period of time.*
/TYPE=*mnem*	Display the file status of only those files that have the same file type mnemonic specified with this switch. Replace *mnem* with a three-character mnemonic, as listed in Table 3.3.

Table 3.9 FILESTATUS conditional command switches

* Enter *[date][time]* in the format *[dt-mon-yy][:hh:mm:ss]*; or time-period in the format *-da[:hh[:mm]]*, where *dt* is the date, *da* is the number of days prior to today, *mon* is the month, *hh* is hours, *mm* is minutes, and *ss* is seconds.

In the *[date][time]* format you can specify the date only, the time only, or both together. If you omit the date, today's date is assumed; if you omit the time, midnight is assumed.

For example, enter *12-JAN-81:10:15:00* for 10:15 a.m., January 12, 1981; or *-1:1* for the last 25 hours.

As already mentioned, you can display information about files in alphabetical order by filename using the /SORT switch. However, you can display the same information sorted in a different way by using the keyword form of this switch, /SORT=*field*. The following table lists the fields on which you can sort.

/Sort=	Sort File List By:
ELEMENTSIZE	Element size — smallest to largest
LENGTH	File length — shortest to longest
TLA	Date last accessed — earliest to latest
TLM	Date last modified — earliest to latest
TYPE	File type

Table 3.10 Sorting file list by fields

As with the /SORT switch, if there is insufficient memory to perform a sort, the CLI will ignore /SORT=*field* and display the file information in unsorted order.

The following examples illustrate use of the FILESTATUS command.

```
) FILESTATUS ↲

DIRECTORY @DPDO:PHOTOG

DARK_ROOM  CAMERAS  SETUP  LIGHTING
```

Entering the command without switches displays the parent directory and the files in that directory.

```
) FILESTATUS/ASSORTMENT/SORT ↲

DIRECTORY @DPDO:PHOTOG

CAMERAS        UDF     30-DEC-80      14:16:30       976
DARK_ROOM      UDF      5-DEC-80      15:14:30      3162
LIGHTING       DIR      1-DEC-80      11:03:12      4608
SETUP          TXT      3-NOV-80      13:11:26      1340
```

The preceding command line displays filename, file type, date and time when the file was last modified, and file length in bytes for each file in the current working directory, @DPDO:PHOTOG. It also sorts this information in alphabetical order according to filename.

```
) FILESTATUS/AFTER/TLM=1-DEC-80/SORT=TLM/ASSORTMENT ↲

DIRECTORY @DPDO:PHOTOG

LIGHTING       DIR      1-DEC-80      11:03:12      4608
DARK_ROOM      UDF      5-DEC-80      15:14:30      3162
CAMERAS        UDF     30-DEC-80      14:16:30       976
```

Requests a listing of the status information for all files modified on or after December 1, 1980. The files are sorted according to the time they were last modified.

4

The CLI Environment & System Parameters

Your interaction with the CLI is affected by a set of parameters whose values define the current CLI environment. This environment can be modified at any time. The CLI PUSH/POP facility enables you to set up, save, and move between environments easily.

Besides modifying parameters that define the CLI environment, you can also alter certain parameters controlling the operating system. Some global system parameters can be changed by the CLI or any other program running in the system without affecting other activity in the system. These parameters control I/O device accessibility, system date, and system time. You can also shut down the current system; however, this can be changed only by the initial CLI.

CLI Environment

The CLI environment directly affects operation of the CLI. It is defined by the following parameters:

- Console characteristics
- Working directory
- Searchlist
- Environment level
- Contents of the string buffers
- Prompt command sequence
- Exception condition responses
- Environment level
- Setting of the mode that measures command execution times
- Setting of command trace mode

(The last two parameters, set by the TRACE and MEASURE commands, are described at the end of Chapter 6.)

Default values for the CLI environment are established at the time the system is generated. Refer to the SYSGEN program in *MP/AOS System Generation and Related Utilities* (DGC No. 069-400206). You can override some of these values by issuing various CLI commands.

Table 4.1 lists the commands you can use to manage the CLI environment and briefly describes the function of each.

Command	Action
CHARACTERISTICS	Set or display the features of the user's console.
CLASS1	Set or display the CLI response to errors caused by commands affecting the CLI environment.
CLASS2	Set or display the CLI response to errors caused by commands not affecting the CLI environment.
CURRENT	Display the current CLI environment values.
DIRECTORY	Set or display the current working directory.
LEVEL	Display the current CLI environment level.
MEASURE	When on, display how much time elapsed during the execution of each CLI command.
POP	Change the current CLI environment level to the next numerically lower level; do not save previous values.
PREVIOUS	Display the previous CLI environment level's values.
PROMPT	Set or display the prompt command sequence.
PUSH	Save the current CLI environment's values, then change the current environment level to the next numerically higher level.
SEARCHLIST	Set or display the contents of the searchlist.
STRING	Set or display the string buffers.
TRACE	When on, display each command line as it appears to the CLI before the CLI executes it.

Table 4.1 CLI environment commands

CLI Environment Levels

The CLI allows you to maintain the state of an environment while moving to another, distinct environment.

Levels are numbered from 0 to *n*, where the maximum level depends on the current use of system resources. When you start out in the CLI, you are in environment Level 0.

When you add a new environment level, its number will be one larger than the previous level. The new level is sometimes called a "higher" level, although the terms "high" and "low" refer only to the level number. Level 0 is the lowest level possible. Figure 4.1 illustrates the environment PUSH/POP mechanism.

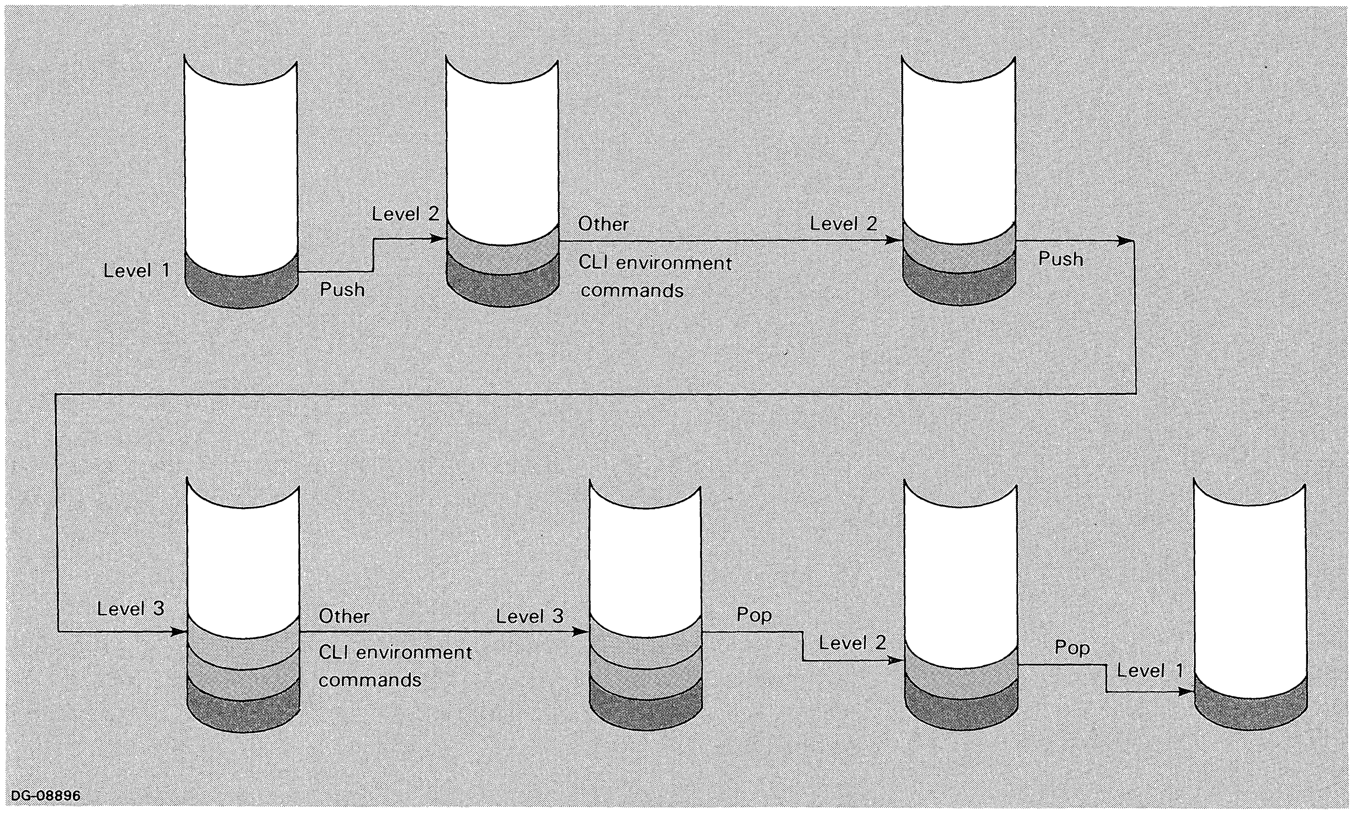


Figure 4.1 Push/pop environment mechanism

Initially, a new level has the same environment values as the level it was created from. However, levels are completely separate. Therefore, you can change the environment values for the new level without affecting the previous level.

When you return from a higher level to a lower level, the higher environment's values are not saved.

The commands used to move from one level to another, and to verify user environment values, are:

LEVEL
PUSH
POP
CURRENT
PREVIOUS

Each of these commands is described in this chapter.

Displaying Current Environment Level

If you want to check at which CLI environment level you are currently processing, issue the LEVEL command.

```
)LEVEL ⤶
```

For example,

```
) LEVEL ⤶
LEVEL 0
```

displays the current CLI environment to be at Level 0.

Do not confuse LEVEL with SWAPLEVEL: LEVEL indicates the current CLI level, and SWAPLEVEL indicates the program execution level.

Changing Environment Levels

The PUSH command enables you to change your CLI environment to a new level while saving the environment values for the level you leave.

```
)PUSH ⤶
```

The new level has the next higher level number. Initially, it has the same environment values as the level from which you PUSH. However, you can change the environment values for the new level at any time.

If you include the /V switch with PUSH, the CLI verifies the new environment level by displaying the level number.

To return to the previous environment level and restore its values, enter the POP command.

```
)POP ⤶
```

This command decrements the current environment level number by one and returns you to that level, restoring its environment values. POP does not save the environment you are leaving.

If you include the /V switch, the CLI verifies the new environment level by displaying the level number.

```
) LEVEL ⤶
LEVEL 0
) PUSH/V ⤶
LEVEL 1
) PUSH/V ⤶
LEVEL 2
) POP/V ⤶
LEVEL 1
) POP/V ⤶
LEVEL 0
```

The above sequence of commands shows you how PUSH increments the level by one and POP decrements the level by one.

```
) LEVEL ⤶
LEVEL 0
) POP ⤶
ERROR: CANNOT POP FROM LEVEL 0
```

Does not POP the current level because Level 0 is the last level to which you can POP.

```
) LEVEL ⤶
LEVEL 0
) DIRECTORY ⤶
:FLUIDS
) PUSH/V ⤶
LEVEL 1
) DIRECTORY :THERMO ⤶
) DIRECTORY ⤶
:THERMO
) FILESTATUS/SORT/TYPE=DIR ⤶
DIRECTORY :THERMO
DENSITY  PRESSURE  TEMP  DENSITY
) POP/V ⤶
LEVEL 0
) DIRECTORY ⤶
:FLUIDS
```

Indicates that the current level is Level 0 with the working directory :FLUIDS. The environment level is then PUSHed to Level 1, and the working directory is changed to :THERMO. The FILESTATUS command displays the directory files subordinate to the working directory (:THERMO). When the level is popped back to Level 0, its working directory is restored (:FLUIDS).

Displaying Current Environment Values

To check what settings exist for the current CLI environment, enter the CURRENT command. The format is simply

```
)CURRENT ↲
```

For example:

```
LEVEL:              0
DIRECTORY:          @DPD0:CHEMO:ORGANIC
SEARCHLIST:         @DPD0:POLYMER
CHARACTERISTICS:    /CPL=80/LPP=24
                    /ON/ST/ECHO/605X
                    /OFF/NAS/ESC/LIST/BIN/UCO/8BT/NED/EMM/ICC
CLASS1:             ERROR
CLASS2:             WARNING
TRACE MODE:         OFF
MEASURE MODE:       OFF
STRING BUFFERS:
PROMPT:
```

Managing Other CLI Environment Values

The environment level is a value set and saved by the PUSH command. PUSH saves many other environment values as well. You set and display these values with specific commands.

Console Characteristics

The CLI environment includes the characteristics of the user's console.

The CHARACTERISTICS command determines how each character device will handle data. It determines such characteristics as how many lines can be displayed at one time, whether or not your input is echoed, and which characters are interpreted literally and which are not.

The CHARACTERISTICS command has this format:

```
)CHARACTERISTICS ↲
```

In this form of the command, you do not specify a device, and the CLI assumes you are referring to the user console. The console is actually two devices, the keyboard and the display, but the CLI displays and sets the characteristics for both at the same time for the user console only.

To set the console characteristics, include the appropriate switches listed in Table 4.2. Enter those you want turned on after the /ON switch. Enter those you want turned off after the /OFF switch. Exceptions are the /CPL=*n* and /LPP=*n* switches, which are not entered with /ON or /OFF. Change the current value for these by entering a different value for *n*. Notice that some of these switches may affect input only, output only, or both input and output.

NOTE: *The /ON switch is optional as long as /OFF does not precede the switches you want to turn on.*

If you enter CHARACTERISTICS without switches, the CLI displays the current characteristics.

Whenever you turn a characteristic on or off, the other characteristics for that device are unaffected. The characteristics you change remain in effect until you change them again, or until you shut down the system.

The initial characteristics for a character device depend on how the device was set up during system generation. See *MP/AOS System Generation and Related Utilities* (DGC No. 069-400206).

To return the console to the initial characteristics for the current version of the CLI, simply enter CHARACTERISTICS/I. To reset it to its SYSGENed settings, specify the /RESET switch. To set it to the characteristics of the previous environment level, specify the /P switch.

NOTE: *The CHARACTERISTICS command can set the characteristics of any character device. However, only the characteristics of the user's console are saved as part of the current CLI environment. For information on use of CHARACTERISTICS with other devices, see Chapter 9, "CLI Command Dictionary."*

Switch	Action
/OFF	Turn off the characteristics named by the switches that follow this switch until the switch list ends or /ON switch occurs in the command line.
ON	Turn on the switches that follow this switch until the switch list ends or /OFF appears. This is optional, because unless a switch you specify follows /OFF, the CLI turns it on.
	The switches that follow are turned on when preceded by /ON and off when preceded by/OFF.
/605X	When on, the Delete key erases the character preceding the cursor, and CTRL-U deletes the entire cursor line. This applies to consoles 6052 and 6053 only. /ECHO must be on.
/8BT	When on, leave all characters as 8-bit characters. When off, mask all input characters to 7 bits.
/BIN	Set Binary mode on or off. When on, permit an 8-bit character to be passed on input and disregard control characters on input and output.
/ECHO	When on, echo all characters read from the input to the output device; interpret special characters and echo control characters by prefixing them with an up-arrow. When echo mode is off, control characters are still interpreted, but it is the programmer's responsibility to echo characters.
/EMM	When on, echo characters exactly as input; control characters are not echoed as ↑x.
/ESC	Set Escape mode on or off. When on, the Escape character is interpreted as a CTRL-C CTRL-A sequence.
/ICC	When on, ignore control characters, except delimiters and characters interpreted by the system.
/LIST	Set List mode on or off. When on, CTRL-L (octal 14) is output as ↑L rather than as a Form Feed.
/NAS	Set non-ANSI standard on or off. When on: input Carriage Returns are converted to Line Feeds, while input Line Feeds are converted to Carriage Returns. Output Line Feeds are echoed as Carriage Return-Line Feed-Null.
/NED	When on, does not echo delimiters.
/ST	Set Software Tab Simulation mode on or off. When on, CTRL-I simulates a tab stop every 8th column on output. Binary Mode (/BIN) must be off.
/UCO	When on, translate lowercase characters to uppercase on output.
	The switches that follow cannot be entered with /ON or /OFF.
/CPL=*n*	Set the maximum number of characters per line to *n* for the specified device.
/LPP=*n*	Set the maximum number of lines per page to *n* for the specified device.
/I	Reset the user console to the characteristics it had when the current version of the CLI began executing. No argument or switches other than standard switches are allowed.
/P	Set the user console to the characteristics it had in the previous environment level. No argument or switches other than standard switches are allowed.
/RESET	Reset the characteristics of the specified device to the values they had at the time the system was booted. No switches other than standard switches are allowed.

Table 4.2 CHARACTERISTICS command switches

In response to:

```
) CHARACTERISTICS ↲
/LPP=24/CPL=80
/ON/605X/ST/UCO/ECHO
/OFF/BIN/ESC/LIST/NAS/8BT/NED/EMM/ICC
```

the CLI displays the current characteristics for the console. Now, turn on the /LIST switch and turn off the /ST switch.

```
) CHARACTERISTICS/ON/ECHO/OFF/ST ↵
```

To set the console's line length to 60 characters, enter

```
) CHARACTERISTICS/CPL=60 ↵
```

Working Directory

Each CLI environment has a working directory. This can be any directory in the MP/AOS file system residing on a mounted disk.

You can display or change the working directory with the DIRECTORY command, as described in Chapter 3.

Searchlist

The searchlist, described in Chapter 3, is also an important part of your environment. It contains a list of directories the system automatically looks through whenever a file requested for access cannot be found in the working directory.

Use the SEARCHLIST command, as described in Chapter 3, to change or display the contents of the searchlist.

CLI String Buffers

There are five string buffers available to each user environment in MP/AOS, and each buffer can contain up to 127 characters. The STRING command displays and sets these buffers.

)STRING *[text-string]* ↵

The optional *text-string* is the information you want the string buffer to contain. Initially, all five buffers are empty. You specify which buffer you want to fill by including the /S=*n* switch, where *n* is a number from 1 to 5. If you do not specify a particular buffer (with /S=*n*), the CLI writes *text-string* into String Buffer 1.

```
) STRING/S=3 YELLOW ↵
```

Sets String Buffer 3 to contain the text string YELLOW.

If you enter STRING without arguments and switches, the CLI displays the contents of String Buffer 1.

```
) STRING ↵
BLUE
```

To clear a buffer of its contents, enter STRING with the /K switch and the /S=*n* switch, where *n* identifies the buffer you want to nullify. If you include just /K, the CLI clears only Buffer 1.

```
) STRING/K/S=4 ↵
```

Clears String Buffer 4 of its contents.

The /P switch sets the buffer you specify with /S=*n* to the contents of the corresponding string in the previous environment level. If you enter /P without /S=*n*, the contents of Buffer 1 is changed to the same contents as that buffer for the previous environment level. The next example illustrates the use of the /P switch.

```
) LEVEL ↵
LEVEL 2
) STRING/S=(1,2) ↵
NEW
AGED
) STRING/P/S=2 ↵
) STRING/S=(1,2) ↵
NEW
OBSOLETE
```

Processing is currently in Environment Level 2. STRING displays the contents of the first and second string buffers, and then changes String Buffer 2 to the same contents as String Buffer 2 in the previous level (Level 1). Now when the contents of the buffers are displayed, String Buffer 2 contains OBSOLETE. The other strings are not affected.

A string buffer is particularly useful for storing a program's termination message for later use, or for substituting a text string in a command line of a macro (see Chapter 6 for a description of textual substitution).

NOTE: *The /S switch for the EXECUTE, and XEQ commands writes program termination messages, if any, only to String Buffer 1, to maintain program compatibility between MP/AOS and AOS, which has only one string buffer.*

CLI Prompt Sequence

Normally, whenever the CLI is ready for your input, it displays the prompt symbol,), only. However, you can instruct the CLI to execute a sequence of CLI commands immediately before displaying every prompt. Issue the PROMPT command:

PROMPT *[command-1 [... command-4]]*↵

Entering this command without arguments and switches displays and then executes the current prompt sequence.

You can specify up to four CLI commands to be included in the sequence. However, DO NOT use macro names or commands requiring arguments.

Initially, there are no commands in the prompt sequence. Once you have entered a prompt sequence, you can clear it by entering PROMPT/K without arguments.

If you change the prompt sequence and then decide you want to use the same one established for the previous environment level, enter PROMPT/P without arguments.

```
LEVEL ↵
LEVEL 1
) PROMPT/P ↵

)
```

The empty line before) indicates that there are no commands in the prompt sequence at level 0.

```
) PROMPT TIME ↵
13:45:10
)
```

Sets PROMPT to display the system time before each command prompt, which it does before the next prompt.

```
) PROMPT/K ↵
)
```

Nullifies the prompt command sequence for the current environment.

CLI Exception Conditions

When you enter a command that is syntactically incorrect or unrecognizable by the CLI, or enter an invalid argument for a command, you create an exception condition. The CLI responds to such a condition according to the type of exception condition and the error response that has been prescribed.

An exception condition can be either CLASS1 or CLASS2. Syntax errors and errors in commands that affect the user environment are CLASS1. Any other errors in commands you enter are CLASS2 exception conditions. A CLASS1 exception condition is usually more serious than a CLASS2 exception condition.

How the CLI resolves a CLASS1 or a CLASS2 exception condition depends on the error response set for the condition. For either CLASS1 or CLASS2, this value can be IGNORE, WARNING, or ERROR. Table 4.3 states the CLI response to each of these values.

Value	CLI Response to Error
IGNORE	Display no message for an exception condition and continue processing, if possible.
WARNING	Display a warning message when an exception condition occurs and continue processing, if possible.
ERROR	Display an error message when an exception condition occurs and discard the input in the command buffer at the time the system encounters the mistake.* If a macro is being processed, that macro (and any macro calling it) will terminate in error.

Table 4.3 Error responses to exception conditions

**The command buffer contains all input from the last prompt to the New-line character terminating the command sequence. This means that at the time the error is detected, the buffer might contain one or more commands and/or macro calls.*

You can set both classes of exception conditions for an entire CLI session (using the CLASS1 and CLASS2 commands), and you can also reset a condition for the duration of a specific command (using the standard command switches /1=*exception-condition* and /2=*exception-condition).*

The CLASS1 and CLASS2 commands are described below, while the /1= and /2= switches are described at the end of this section on "CLI Exception Conditions."

The initial response value for CLASS1 is ERROR, and for CLASS2, WARNING.

The CLASS1 command displays and sets the error response for all CLASS1 exception conditions that occur in the current environment. The command format is

)CLASS1 *[error-response]* ⤶

Replace *error-response* with one of the values listed in Table 4.3. If you enter CLASS1 without an argument, the CLI displays the current error-response value. For example,

```
) CLASS1 ⤶
ERROR
) CLASS1 WARNING ⤶
```

displays the current CLASS1 setting, ERROR, and then changes it to WARNING.

The CLASS2 command displays and sets the error response for all CLASS2 exception conditions that occur in the current environment. Its format is similar to CLASS1:

)CLASS2 *[error-response]* ⤶

If you enter CLASS2 without an argument, the CLI displays the current error-response value for CLASS2 exceptions.

```
) CLASS2 ↲
WARNING
) CLASS2 IGNORE ↲
```

Displays the current error response as WARNING, and then changes it to IGNORE.

The next several examples illustrate how the different error responses affect processing of multi-command entries.

```
) CLASS1 ↲
ERROR
) SEARCHLIST :POLYMER;TIME ↲
ERROR: FILE DOES NOT EXIST
SEARCHLIST,:POLYMER
```

Displays the current value for CLASS1. SEARCHLIST causes a CLASS1 error because the system cannot find the :POLYMER directory. Therefore, the error message is displayed and the TIME command is discarded by the CLI.

```
) CLASS1 WARNING ↲
) DIRECTORY BENZINE;LEVEL ↲
WARNING: FILE DOES NOT EXIST
DIRECTORY,BENZINE
LEVEL 0
)
```

Changes the original CLASS1 value from ERROR to WARNING. The DIRECTORY command causes an error, displaying a warning message this time. Because the error response is now WARNING, the CLI continues processing the command line and displays the current CLI environment level.

```
) CLASS2 ↲
WARNING
) TYPE METHANOL;DELETE/V ALCOHOLS ↲
WARNING: FILE DOES NOT EXIST, FILE METHANOL
DELETED ALCOHOLS
```

Displays the current CLASS2 error response as WARNING. Since the METHANOL file does not exist, the warning message is displayed. The CLI continues processing the command line by deleting ALCOHOLS.

For the next example, assume that file AMINO already exists.

```
) CLASS2 IGNORE ↵
) RENAME BIO_SYN AMINO;TIME
15:10:30
)
```

Changes the value for CLASS2 to IGNORE. Because AMINO already exists, the RENAME command causes a CLASS2 exception condition. No error message is displayed, however, because of the CLASS2 setting. Neither BIO_SYN nor AMINO is changed. The CLI continues processing by displaying the current system time.

You can override the current error response setting for the duration of a command by including the /1=*error-response* standard switch for a CLASS1 exception condition and/or the /2=*error-response* standard switch for a CLASS2 exception condition. Replace *error-response* with IGNORE, WARNING, or ERROR. For example,

```
) CLASS2 ↵
IGNORE
) RENAME/2=WARNING REPORT MILESTONES ↵
WARNING: FILE ALREADY EXISTS, FILE MILESTONES
) CLASS2 ↵
IGNORE
)
```

displays the current setting for CLASS2 to be IGNORE. This is temporarily changed to WARNING for the RENAME command entry. An exception condition occurs while RENAME is being processed; hence the CLI displays the warning message. Notice that CLASS2 is still set to IGNORE.

Displaying Previous Level's Environment Values

If you have changed the level of your CLI environment and are currently running at a level other than Level 0, you can take a look at the environment settings for the previous level with the PREVIOUS command. PREVIOUS is simply entered as

```
)PREVIOUS ↵
```

This command does not change the current environment level or its settings. If you enter this command while in Level 0, it will return an error.

```
LEVEL:            0
DIRECTORY:        :THERMO
SEARCHLIST:       :MATERIALS,:HEAT
CHARACTERISTICS:  /CPL=80/LPP=24
                  /ON/ST/ECHO/605X
                  /OFF/NAS/ESC/LIST/BIN/UCO/8BT/NED/EMM/ICC
CLASS1:           ERROR
CLASS2:           WARNING
TRACE MODE:       OFF
MEASURE MODE:     OFF
STRING BUFFERS:
                  EQUILIBRIUM


PROMPT:           TIME
```

Displays the current environment level as Level 1, and then displays the environment settings for the previous level, Level 0.

```
) POP/V ↵
LEVEL 0
) PREVIOUS ↵
ERROR: NOT VALID FROM LEVEL 0
```

Returns to Level 0 and attempts to list the environment for the previous level. Since there is no level previous to Level 0, the CLI displays an error message.

System Parameters

As already mentioned earlier in this chapter, there are a few system parameters that you can modify from the CLI. These parameters affect disk accessibility, the system time, and the system date.

More drastically, from the initial CLI process you can use one CLI command to shut down the operating system.

The CLI commands that modify these system parameters are listed in Table 4.4.

The remainder of this chapter describes how you can use these commands to modify the system parameters.

Command	Action
BOOT*	Shut down the current system
DATE	Set or display the system date
DISKSTATUS	Display disk information
DISMOUNT	Remove a disk device from the system
MOUNT	Make a disk file's structure available to the system
TIME	Set or display the system time

Table 4.4 Commands affecting system parameters

**Restricted to initial process. Note that either BYE or CTRL-C CTRL-B, when issued from the initial process, will also shut down the system.*

Mounting a Disk Device

Except for the system master device, from which MP/AOS is initially loaded, you must explicitly make a disk's directory structure available to the system before you can access the files on them. Use the MOUNT command to perform this operation.

)MOUNT *directory-device [disk-id]* ⤶

Replace *directory-device* with the unit name of the disk or diskette you want mounted. Table 4.5 lists the disks that are available to the MP/AOS system.

The second argument, *disk-id*, is optional but can be included to verify that you are mounting the right device. If you specify this argument, the system checks the device to see if it has a matching ID. If the ID does not match, the disk is not mounted. Whenever a device is successfully mounted, the CLI confirms this by displaying the disk ID. (You assign a disk ID when you software initialize the disk using the DINIT utility, described in *MP/AOS System Utilities Programmer's Reference,* DGC No. 069-400206.)

Unit Name	ECLIPSE	microNOVA
DPG	20 Mbyte cartridge disk	
DPD	10 Mbyte cartridge disk 315 Kbyte diskette	10 Mbyte cartridge disk
DPH	12.5/25 Mbyte fixed disk 1.25 Mbyte diskette	12.5/25 Mbyte fixed disk 1.25 Mbyte diskette
DPF	50/95/190 Mbyte cartridge disk	
DPX		315 Kbyte diskette

Table 4.5 Disk and diskette unit names

Suppose you want to access disk @DPD3, which is not mounted in the system. Typing

```
) MOUNT @DPD3 ⤶
MY_DISK
```

makes the files on the disk available, and confirms that it has been mounted by displaying the disk's ID, MY_DISK.

```
) MOUNT @DPD4 DISK_1 ⤶
DISK_1
```

This example checks the @DPD4 disk device for a label named DISK_1, and then confirms it as the requested device. The files on the disk can now be accessed.

Dismounting a Disk Device

If you want to unMOUNT a disk, use the DISMOUNT command.

) DISMOUNT *directory-device* ⤶

The disk or diskette that you specify as *directory-device* is disabled from further file system activity, and is prepared for removal from the system without loss of data.

```
) DISMOUNT @DPD4 ⤶
```

Prepares disk @DPD4 for removal from the system.

NOTE: *The system master device and all other disk devices are automatically dismounted when you shut down the system.*

Displaying Disk Statistics

You can display information about space available on a disk and about the disk's error history by issuing the DISKSTATUS command.

)DISKSTATUS *[diskname]* ⤶

If you do not specify a *diskname,* the CLI displays information about the disk containing the working directory. This command displays the number of disk blocks available and the number used. Including /B lists this same information in bytes instead of blocks. To find out how many files can be stored on the disk, include the /F switch.

Setting or Displaying System Date

Both the system date and system time (described next) should be set when the operating system is loaded. Usually these values correspond to the actual calendar date and time of day. They may be reset at any time from the CLI.

These values are used to record the time and date a file is created, modified, or accessed. You can use this information yourself (as with the FILESTATUS command) to manage your files. Format the DATE command as

DATE *[dd-mon-yy | dd mon yy | mm/dd/yy]* ⤶

Entering this command without an argument string displays the current system date. To set the date, enter the date in one of the above formats with these substitutions:

dd	one- or two-digit day
yy	two-digit year
mon	first three letters of month name
mm	two-digit month

No matter which format you use to enter the date, the system displays it as *dd-mon-yy*.

```
) DATE 3/25/81 ↓
) DATE ↓
25-MAR-81
```

Sets and then displays the system date.

Setting or Displaying System Time

The TIME command sets or displays the current system time. Use this format:

TIME *[hh:mm:ss]* ↓

where *hh* is hours, *mm* is minutes, and *ss* is seconds. If you enter TIME without an argument, the CLI displays the current system time.

```
) TIME 15:22:19 ↓
) TIME ↓
15:22:23
```

Sets and displays the current system time.

Shutting Down the System

Initially, you must load the MP/AOS system into operation before you can process in the CLI. Once the system is running, it automatically executes the initial CLI. The initial CLI is the initial process in the system; see the description of the initial process in Chapter 5.

To shut down the operating system, enter the BOOT command. BOOT is a restricted command; it can only be issued from the initial process. Its format is

```
)BOOT ↓
```

Before the system is shut down, BOOT closes all open I/O channels and dismounts all disk devices.

NOTE: *Issuing BYE or CTRL-C CTRL-B from the CLI running at Swap level 1 in the initial process will also shut down the system.*

Program & Process Management

5

A process is a memory-resident environment within which a program can execute in parallel with other programs. The CLI program executes within its own process, and provides commands to create and manage other processes.

Although each program executes within a process, you do not have to create a new process for each program you wish to run. Rather, you can execute different programs sequentially within the address space of a single process.

The CLI provides commands to start up new programs within the same process. The state of the CLI for that process can either be saved on disk, or overwritten.

This chapter provides a brief overview of process concepts, including the definition of an MP/AOS process and a brief look at the way memory is allocated to a process. For more detailed information on these issues, refer to the *MP/AOS System Programmer's Reference* (DGC No. 093-400001).

Following this overview, the chapter covers program management from the CLI, and concludes with the CLI process management commands.

Process Concepts

A process is an environment within which a program can execute in parallel with other programs. The process defines and controls the real-time resource management performed by MP/AOS for its users. These resources include the following:

- the program to be executed for this process
- a message which can be passed to the process
- pathnames of devices or files to be open on Channels 0 and 1
- an initial searchlist setting
- an initial directory setting
- initial process priority
- maximum number of channels the process can use
- maximum number of tasks the process can execute
- maximum amount of memory the process can use
- maximum number of memory segments the process can attach to
- maximum number of overlay nodes

The process thus represents facilities allocated to the executing program, as well as constraints within which that program is confined. The constraints indicate the kind and number of system resources required for the execution of a given process, and whether or not the system is capable of executing it.

You can initiate a maximum of 16 individual processes concurrently (or less, depending on the maximum number specified during system generation). Any program can create other processes.

MP/AOS processes are *concurrent* and *resident.* All newly created processes run at Program Swap Level 1, and each process resides in memory. The system allocates CPU control to each process according to process *priority.* Priority is user-assigned.

CLI commands can initiate processes and define their environments, communicate with other processes by means of IPC (interprocess communication) messages, and retrieve process information. From the CLI, you can execute a separate process debugger that can initiate and block a user process for the detection of potential runtime errors during program development.

The Initial Process

The initial process is created by the system when MP/AOS is initialized. The specific program executed by the initial process is a system generation parameter; you can specify any .PR file for this purpose. Typically, the CLI runs as the initial process. The CLI running as the initial process (at Swap Level 1) is known as the *root* CLI.

The initial process is the only process capable of issuing a BOOT command to shut down the system. Terminating the root CLI in Program Swap Level 1 of the initial process (by entering the BYE command) also causes the system to shut down.

Concurrent Processes

MP/AOS does not have a process hierarchy. Instead, all processes are created in parallel and run concurrently, initially at Program Swap Level 1.

Figure 5.1 illustrates multiple processes running concurrently at Program Swap Level 1.

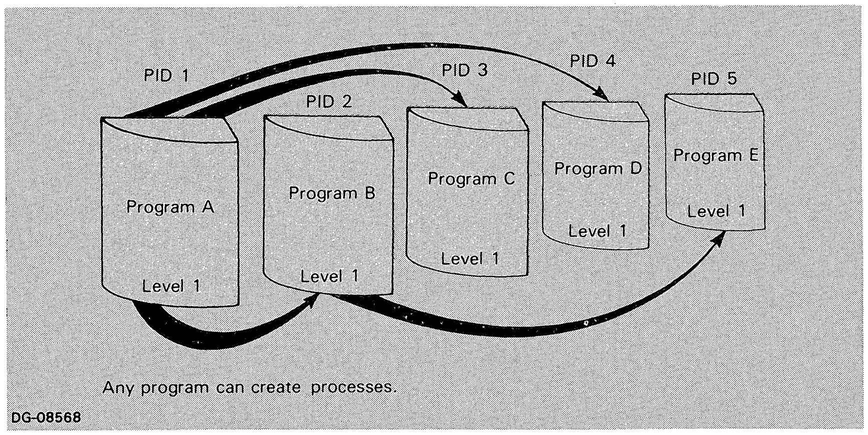


Figure 5.1 MP/AOS concurrent processes

Once a process is created, it has no further relationship or dependency on the process creating it, neither during its life nor as it terminates (except that any process can terminate any other process).

PID Numbers and Program Swap Levels

When a process is created, the system assigns it a *process identity number,* or PID. The PID is an integer in the range 0 to 65535. While the process exists in the system, use this identifier when entering process management commands.

You can replace the CLI program temporarily by executing (EXECUTE or XEQ) another program within the same process. The CLI is then swapped out to disk, and the new program receives a higher *program swap level* number. The PID does not change.

If, instead, you execute the new program with the CHAIN command, the currently executing CLI is not saved on disk. The new program executes at the same program swap level as the CLI it replaced. Again, the PID does not change.

NOTE: Do not confuse the program swap level with the CLI environment level described in Chapter 4.

Resident Processes

All processes are *resident,* that is, they reside in main memory until terminated. Resident processes cannot be displaced by any other process even if performing lengthy I/O or when blocked for an appreciable amount of time.

Process Address Space

When creating a process, you can specify the maximum logical address space it may use in system memory, up to 32 K-words. (A program may or may not use the maximum space allowed, depending on the program's memory requirements as determined by the Binder.) This processing area consists of three memory segments: impure (Segment 0), shared (Segment 1), and overlay (Segment 2). Each segment consists of a number of 1 K-word units called *pages.*

The impure and overlay segments contain information used only by the specified program. Neither is shared with other programs. The shared memory, however, is available to all programs in the system.

The executing program's overlay code is written into the overlay segment. The shared and overlay areas together make up what is called the "pure memory area." Figure 5.2 shows how process address space is configured.

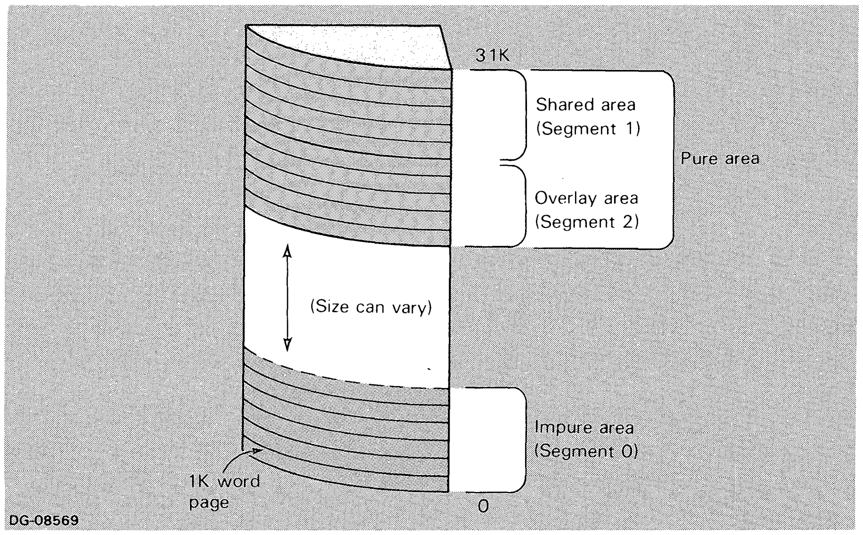


Figure 5.2 Process address space

The maximum size of the total address space is defined at the time you create the process. The size of the shared and overlay areas remains the same during the life of the program. The impure area, on the other hand, may expand or contract as required by the executing program.

Process Priority

Because each process is independent of other processes in the system, each competes for available system resources according to process priority.

Priority is a number from 0 to 255, where 0 is the highest priority and 255 the lowest. You assign this priority to the process when creating it.

Program Management

You do not have to create a new process for each program you wish to execute. You can run different programs within the same address space where your CLI executes. The CLI provides several commands you can use to manage programs.

The CLI enables you to execute or debug a program, display information about a program's type, or change a program's revision number. The commands for managing your programs are listed in Table 5.1.

Command	Action
BYE	Terminate the current CLI
CHAIN	Overwrite the CLI with new program
DEBUG	Invoke the Symbolic Debugger to debug a program
EXECUTE	Execute a program at next program level
INFORMATION	Display program characteristics
REVISION	Display and set the current program revision number
SWAPLEVEL	Display the program swap level of the current CLI
XEQ	Same as EXECUTE

Table 5.1 Program management commands

Changing Programs within a Process

You can execute other programs in a process besides the CLI invoked at process creation time. To do this, you use the EXECUTE or XEQ command to perform a swap operation, or the CHAIN command to do a chain operation. If Channels 0 and 1 are open, they will be passed to the new program. Unless the /I=*pathname* or /O=*pathname* switches are specified, the two channels are passed unchanged from the old program to the new one.

Swapping Programs

The EXECUTE and XEQ commands function identically. Either one will invoke another program in the same process in which you are currently executing the CLI by swapping the CLI out to disk. The program swap level is incremented by 1, and the new program then executes at the incremented program level. This swap operation is shown in Figure 5.3, where the CLI executing in PID Y (any numbered process) is swapped to disk to execute the EMULATE.PR program.

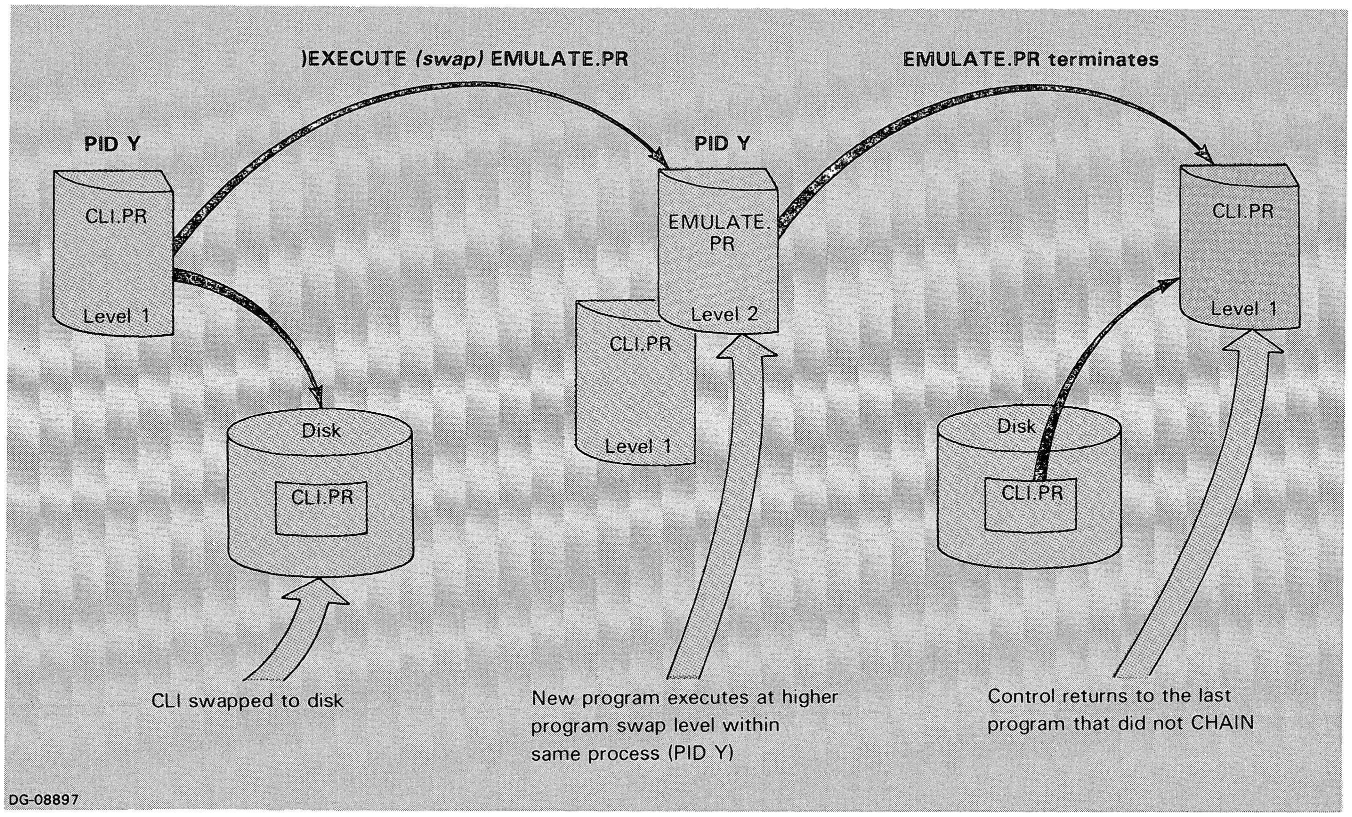

)EXECUTE (swap) EMULATE.PR
EMULATE.PR terminates
PID Y
CLI.PR
Level 1
PID Y
EMULATE.PR
Level 2
CLI.PR
Level 1
CLI.PR
Level 1
Disk
CLI.PR
Disk
CLI.PR
CLI swapped to disk
New program executes at higher program swap level within same process (PID Y)
Control returns to the last program that did not CHAIN
DG-08897

Figure 5.3 Swapping out the CLI with EXECUTE

In Figure 5.3, the CLI is the "parent" of EMULATE.PR. The term parent program refers to the calling program, but has no hierarchical connotations. That is, a parent program is merely the program on the next numerically lower program swap level to the current program.

You can nest programs executing within the process up to eight levels. All programs except the currently executing one are temporarily swapped to disk. Hence, EMULATE.PR in Figure 5.3 could swap itself to disk to execute another program, too. (This new program would run at Program Swap Level 3.)

When the swapped-to program terminates, processing returns to its parent program. Therefore, when EMULATE.PR completes execution, the CLI is swapped back to execute again at its original program swap level, Level 1.

EXECUTE and XEQ are formatted in the same way and accept the same switches.

EXECUTE *program-name [program-arguments]* ⤶

or

XEQ *program-name [program-arguments]* ⤶

Any arguments you enter are passed to the specified program.

```
) EXECUTE BIND PRG_TEST.OB ⤶
```

This command invokes the Binder to bind the object module PRG_TEST.OB. If the CLI is executing at Level 1, the Binder will execute at Level 2. The Binder executes without interruption until it terminates, at which point processing control is returned to the CLI.

Another way to pass input to the new program is to include the /I=*pathname* switch, where *pathname* is an existing *script file.* A script file contains instructions to be entered in place of console input to achieve specific results without keyboard intervention. (If you specify a simple filename, and it is not found through the searchlist or in the working directory, the CLI returns an error message.)

To write program output to a file instead of to the console, include the /O=*pathname* switch. (If you specify a simple filename, and it is not found through the searchlist or in the working directory, the file will be created in the working directory.)

```
) XEQ/I=RESOURCES/O=TIMETABLE PROJECT_PLAN.PR ⤶
```

The command line invokes the program named PROJECT_PLAN.PR, using the RESOURCES file for input. The program will write its output to the file TIMETABLE, rather than to your console.

If you issue EXECUTE (or XEQ) with the /M switch from within a macro, the new program accepts as input the macro lines following the EXECUTE (XEQ) command line. The last line to be read as input to the program must be a single). (See "Using the /M Switch in a Macro", in Chapter 6.)

If you include the /S switch, the program termination message, if any, is written to String Buffer 1 instead of to the console. (The switch cannot be used with /L=*pathname.*)

```
) XEQ/S CONSTRUCT.PR ⤶
```

executes program CONSTRUCT.PR and writes the program termination message (if any) to String Buffer 1 for later use.

Displaying the CLI Program Swap Level

To display the program swap level at which the current CLI is running, issue the SWAPLEVEL command. Because an executing program can invoke the CLI, it is possible for the CLI to run at some level other than Level 1.

Do not confuse SWAPLEVEL with LEVEL: SWAPLEVEL indicates the program execution level, while LEVEL indicates the current CLI environment level. (Chapter 4 describes the CLI environment.)

Chaining Programs

The CHAIN command also allows new programs to execute within a process address space. However, instead of swapping the CLI out to disk to continue processing later, CHAIN overwrites the CLI, and none of the information about the state of the CLI that was running is saved. Hence, the new program executes at the same program level as the CLI. This "chain operation" is illustrated in Figure 5.4.

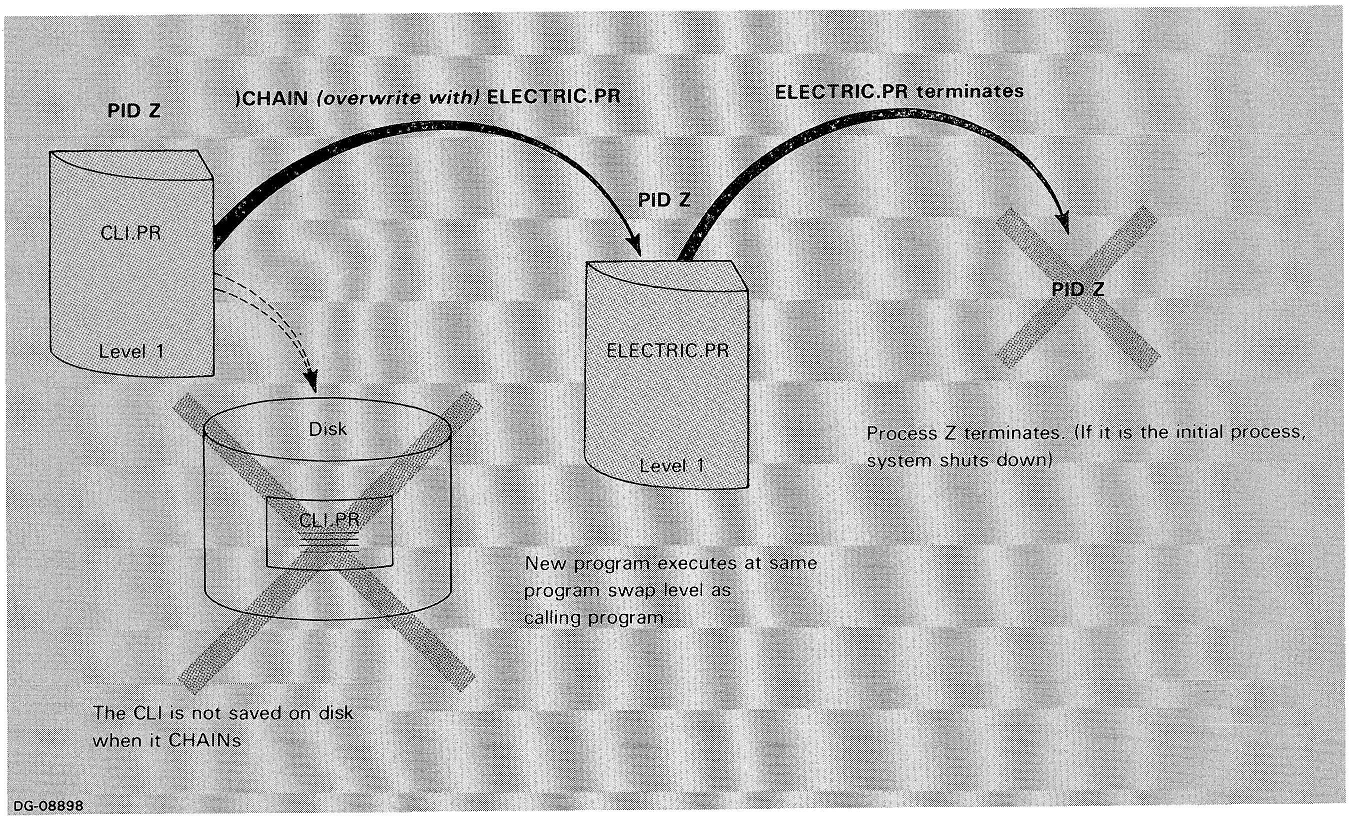

PID Z
)CHAIN (overwrite with) ELECTRIC.PR
ELECTRIC.PR terminates
CLI.PR
Level 1
PID Z
ELECTRIC.PR
Level 1
PID Z
Disk
CLI.PR
Process Z terminates. (If it is the initial process, system shuts down)
New program executes at same program swap level as calling program
The CLI is not saved on disk when it CHAINs
DG-08898

Figure 5.4 Overwriting the CLI with CHAIN

In this figure, CHAIN overwrites the CLI with ELECTRIC.PR. The format for CHAIN is

CHAIN *program-name [program-arguments]* ↵

As with EXECUTE and XEQ, any arguments you specify are passed to the new program. CHAIN supports the /I=*pathname* and /O=*pathname* command switches; these operate in the same way as for EXECUTE and XEQ.

If the new program does not execute another chain operation while it is processing, MP/AOS returns CPU processing control to the last non-chained program upon termination. If there is no previous non-chained program, the process terminates. If the initial process is terminated, the system shuts down. See Figures 5.3 and 5.4 for a graphic view of terminations after XEQ and CHAIN, respectively.

Terminating Programs

You can end execution of any program in the system, regardless of program swap level, with the TERMINATE command. (Also, if you shut down the system, all the processes in the system terminate.)

) TERMINATE *process-id* ↵

Note that you specify a PID number for the program you wish to terminate. If the program is running at Program Swap Level 1, the process will terminate as well. Otherwise, the last non-chained-from program for that process will resume execution. Refer to Figure 5.5 for an illustration of these effects.

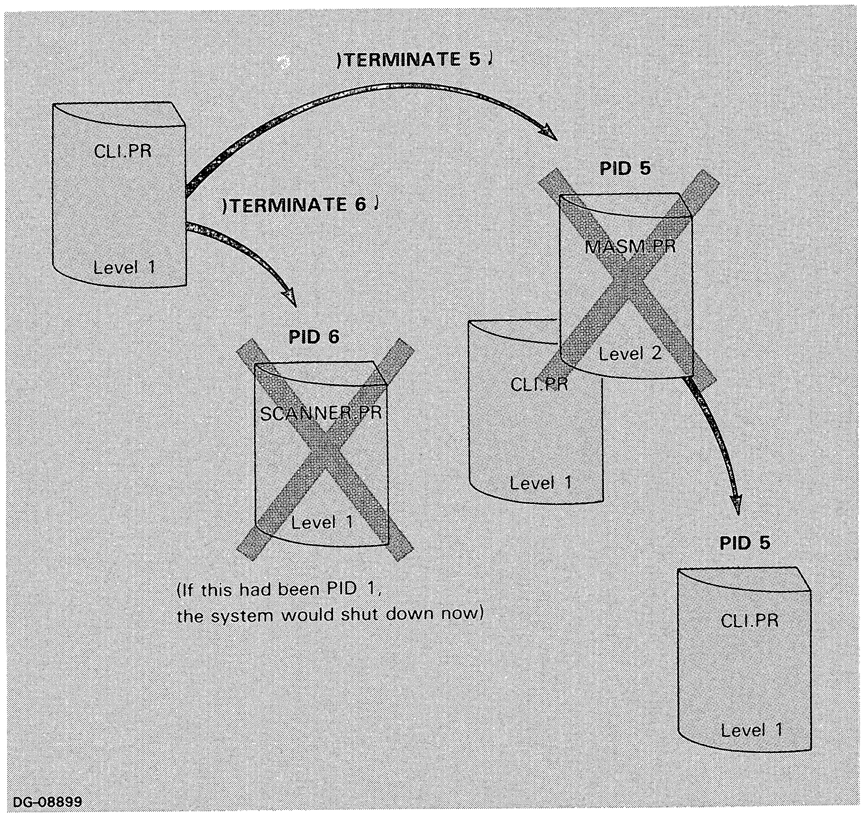


Figure 5.5 Effects of the terminate command

You can terminate your own CLI by specifying its PID (but it's easier to use the BYE command to terminate the CLI).

If you wish, you can create a break file when the program terminates. A break file saves the current processing values of the executing program. To create a break file, include the /BREAK switch. MP/AOS assigns the default name ?*pid.time*.BRK, where *pid* is the PID number of the process and *time* is the current time in the form *hh_mm_ss* (two-digit hours, minutes, and seconds). Later, you can peruse this file.

For example, to end PID 5 and save its current values in a break file, type

```
) TERMINATE/BREAK 5 ↲
```

Terminating the CLI

To terminate the CLI, use the BYE command.

) BYE *[argument-list | message]* ↲

If you are executing the CLI at a program level other than Level 1, BYE terminates the current CLI and returns control to the last non-chained program. Including a list of arguments or a message with BYE passes it to the last non-chained program.

Entering BYE when the CLI is at Swap Level 1 in the initial process shuts down the MP/AOS system.

```
) BYE ↵
MP/AOS CLI TERMINATING
SYSTEM SHUTDOWN
```

The following example terminates the current version of the CLI and returns the argument list LOADER,EXTENDER to the parent program.

```
) BYE LOADER,EXTENDER ↵
```

At the beginning of the next example, the CLI is processing at Swap Level 2. BYE terminates the CLI and returns processing to the parent program, which happens to be another version of the CLI. This time when the SWAPLEVEL command is issued, the CLI displays that it is executing at Program Level 1.

```
) SWAPLEVEL ↵
LEVEL 2
) BYE ↵
MP/AOS CLI TERMINATING

) SWAPLEVEL ↵
LEVEL 1
```

Debugging Programs

MP/AOS has a utility called the Symbolic Debugger that enables you to monitor and modify program execution. Refer to *MP/AOS Debugger and Performance Monitoring Utilities* (DGC No. 069-400205), for more about the Debugger.

To set up a separate process environment for the program to be corrected, include the /DEBUG switch with the PROCESS command as described earlier in this chapter in "Creating Processes."

If you don't need to create a new process, enter the DEBUG command to invoke the Debugger:

) DEBUG *program-name [program-arguments]* ↵

The program you specify must be an executable program file. For example,

```
) DEBUG HI_TECH.PR ↵
```

executes the HI_TECH.PR program and invokes the Symbolic Debugger to debug the program. MP/AOS blocks the program before it begins execution, allowing the Symbolic Debugger to examine or modify its state.

Displaying Program Information

Use the INFORMATION command to display the type of any of your programs.

```
) INFORMATION program-pathname ↲
```

The command accepts a pathname or simple filename as an argument. It displays the kind of file the program is, such as an overlay file.

Command switches display additional information about the program. The /I switch indicates the starting address and length of impure memory (Segment 0). The /P switch displays the starting address and length of pure memory, which consists of shared memory (Segment 1) and overlay memory (Segment 2). The /R switch displays the program revision number.

The Program Revision Number

You can display or set the revision number for a program with REVISION.

```
) REVISION pathname [major.minor] ↲
```

Major and *minor* are the major and minor revision numbers, which can range from 0 to 255. The revision number enables you to mark and keep track of different versions of the same program.

```
) REVISION COUNT.PR 00.03 ↲
) REVISION COUNT.PR ↲
00.03
```

sets and displays the revision number for COUNT.PR.

If you include /V, the CLI displays the program name with the revision number.

Process Management

Sometimes you don't want to make the CLI inaccessible while you execute another program. When this is the case, you can start up another process to run concurrently with the CLI.

The CLI enables you to create and terminate a process, retrieve status information about a process or the system, communicate with other processes, and control process scheduling. The CLI commands that perform these functions are listed in Table 5.2.

Command	Action
ACTIVE	Display current list of PID numbers in use
BLOCK	Stop a process
CONTROL	Send an interprocess communication message to a server process
PRIORITY	Return process priority of specified process
PROCESS	Create a process
RUNTIME	Report resource usage statistics
SYSTEM	Display the status of the system
TERMINATE	End a process
UNBLOCK	Restart a blocked process
WHO	Identify a process

Table 5.2 Process management commands

The remainder of this chapter discusses how to manage processes using these commands.

Creating Processes

To create another program in a separate process to run concurrently with the CLI, use the PROCESS command.

) PROCESS *program-name [argument-1 [...argument-n]]* ↲

The new process will begin immediately to execute the specified program at Program Swap Level 1. The system assigns the new process a process identity number (PID).

Arguments are passed to the program as it begins execution.

Each process is a separate environment. Figure 5.6 illustrates the relationship between the system environment, process environment, and CLI environment. As you may remember, the process environment specifies the conditions under which the program executes. You can define some environment parameters for new processes by including PROCESS command switches when creating the process. The process environment for the initial process is set up at system generation.

In addition, you can use a switch to invoke the Symbolic Debugger when you create the process. The Symbolic Debugger invokes the program to be executed.

Table 5.3 shows the PROCESS command switches.

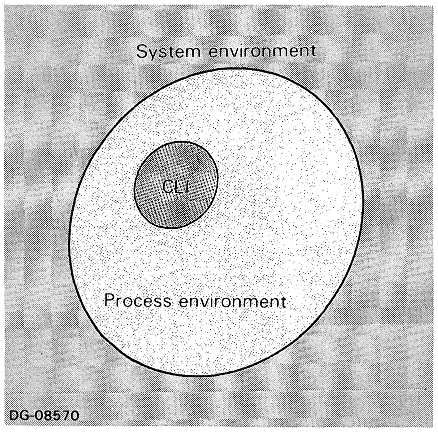


Figure 5.6 System, process, and CLI environments

Switch	Action	Process Value if Switch Omitted
/CHANNELS=*n*	Specify maximum number of I/O channels process can open.	Same as CLI.
/DEBUG	Invoke FLIT with program.	Program executes without FLIT.
/I=*pathname*	Pre-open the file or device specified in *pathname* for input on Channel 0.	Nothing pre-opened on Channel 0.
/O=*pathname*	Pre-open the file or device specified in *pathname* for output on Channel 1.	Nothing pre-opened on Channel 1.
/OVNODES=*n*	Specify maximum number of overlay nodes process can have.	Same as CLI.
/PAGES=*n*	Set the maximum size of process memory space, in 1 K-word pages.	Same as CLI.
/PRIORITY=*n*	Set priority of new process.	Same as CLI.
/SEGMENTS=*n* *	Specify the maximum number of memory segments that can be attached to the program.	Same as CLI.
/TCBS=*n*	Specify maximum number of task control blocks (and therefore, tasks) the process can create.	Same as CLI.

Table 5.3 Process command switches

**Does not include size of impure, shared, and overlay areas allocated by /PAGES=n.*

Set the process priority with /PRIORITY=*n*, where *n* is in the range 0 (highest priority) to 255 (lowest priority).

The switch /PAGES=*n* sets the maximum size of process memory space, which consists of impure, shared, and overlay segments. You can specify a total memory area of up to 32 pages. Future programs executing within this process cannot be larger than the specified maximum number of pages.

The /SEGMENTS=*n* switch specifies the maximum number of memory segments that can be attached to the program after the initial address space has been allocated. These segments can be any size and are separate from the address space. Attached segments are useful for reading and writing data between processes.

You can also specify the maximum number of overlay nodes and task control blocks (TCBs) for the process by including /OVNODES=*n* and /TCBS=*n*, respectively. The program can create one task for each TCB allowed the process, up to a maximum of 255 TCBs.

Data transfers between a process and a device occur by way of an I/O channel. To set the maximum number of I/O channels a new process can open, include the /CHANNELS=*n* keyword switch.

If the program to be executed in the new process expects Channel 0 (input) and Channel 1 (output) to be open when it begins execution, you must pre-open them by specifying /I=*pathname* (for input) and /O=*pathname* (for output). (Note that this is unlike XEQ and CHAIN, where the program created is in the same process, and Channels 0 and 1 *are* passed to the new program.)

The two remaining process environment parameters are the searchlist and working directory. MP/AOS uses these to locate the files your program will access. The current CLI searchlist and working directory are passed to the new process. These parameters can be modified by the program itself after it is started.

For your convenience, the environment values for the initial process (which executes the root CLI) are listed in Table 5.4.

Parameter	Value
Priority	0
Memory	
Maximum number of 1 K-word pages	32 pages
Maximum number of attached segments	0 (segments)
Maximum number of overlay nodes	2 (nodes)
Tasks	
Maximum number of TCBs	3
Channels	
Input channel	@TTI
Output channel	@TTO
Maximum number of channels	15
Searchlist	:
Working Directory	:

Table 5.4 Environment values for the initial MP/AOS process

Pausing Processes

You can stop a process by using the BLOCK command.

) BLOCK *process-id* ⤶

The process you BLOCK will not run until another process restarts it with the UNBLOCK command. If the process is already blocked, BLOCK will have no effect. Entering

) UNBLOCK *process-id* ⤶

resumes execution of a blocked process. If the process is not blocked, UNBLOCK will have no effect.

Process Status & Communications

The CLI provides a number of commands that supply process status information, namely the ACTIVE, WHO, RUNTIME, PRIORITY, and SYSTEM commands. In addition, a command is available for interprocess communication: CONTROL.

Process Identification

To get the process identity numbers (PID numbers) of all executing (or blocked) processes, use the ACTIVE command:

```
) ACTIVE ⤶
```

The command returns the PID number of all active processes. For example:

```
ACTIVE ⤶
ACTIVE PIDS:
1
6
7
23
)
```

For information about an individual process, type the WHO command.

) WHO *[process-id]* ⤶

This command displays the PID and the name of the program the process is currently executing.

```
) WHO 7 ⤶
PID: 7    @DPDO:UTIL:MASM.PR
```

shows that Process 7 is executing the Macroassembler.

If you do not enter a PID number, MP/AOS displays the information for the CLI.

```
WHO ⤶
PID: 9    @DPHO:CLI.PR
```

The current process in this example is PID 9 executing the CLI.

A very useful macro combines the WHO command with ACTIVE used as a textual substitution. The ?.CLI macro contains the one line:

```
WHO [!ACTIVE]
```

Execution of this macro provides status information for all processes. For example:

```
) ? ⤶
PID: 1    @DPDO:CLI.PR
PID: 2    @DPDO:MYPROGS:MEASURE_IT.PR
PID: 23   @DPDO:CLI.PR
)
```

Process Resource Statistics

RUNTIME reports statistics about a process' use of system resources, including

elapsed time since the process was created
CPU time used
number of blocks of data read or written
number of characters transferred

Enter this command as

```
) RUNTIME [process-id] ↓
```

As with WHO, if you do not specify a process, the statistics are reported for the currently executing CLI.

To find out the status of PID 15, enter

```
) RUNTIME 15 ↓
ELAPSED 21:44:09, CPU 0:01:47.009, I/O BLOCKS 927, CHARACTERS 93901
```

The response indicates that the process has existed in the system for 21 hours, 44 minutes, 9 seconds; utilized 1 minute, 47.009 seconds CPU time; read and written 927 blocks of data; and transferred 93901 characters.

To discover the priority of any process, use the PRIORITY command.

```
) PRIORITY [process-id] ↓
```

You supply a PID as an argument to the command.

```
PRIORITY ↓
3
)
```

The priority of the CLI is 3 (third from the highest).

System Information

To find out the maximum number of concurrent processes allowed by the system, type SYSTEM. (The maximum number of processes is set during system generation; at most a system can allow 16 concurrent processes.) This command also displays the system revision number, the amount of physical memory used by the system, and the number of pages free in memory. Including the /ID or /DISK switch displays the system name and the disk from which the system was booted, respectively.

For example,

```
) SYSTEM ↓
SYSTEM INFORMATION
   REVISION: 1.00
   PHYSICAL MEMORY PAGES IN USE: 120
   PAGES AVAILABLE: 136
   MAXIMUM CONCURRENT PROCESSES: 5
)
```

Communicating between Processes

To communicate with programs running within other processes, you can send an interprocess communication (IPC) with the CONTROL command.

To send a message via the interprocess communication (IPC) facility, the destination process must already be defined as a server capable of receiving such a message. (See Chapter 7 in the *MP/AOS System Programmer's Reference* (DGC No. 093-400051), for an explanation of interprocess communication.) You send the message with the CONTROL command.

```
) CONTROL ipc-server-name message ↓
```

The CLI sends the message to the specified IPC server.

Macros, Pseudomacros & Textual Substitutions

6

A macro is a file containing a command or argument sequence that you can use to save time and effort. A frequently-used command sequence can be made into a macro and later executed by simply entering the macro name.

If you include dummy argument references in your macro, you can use the macro for more than one application.

You can further enhance your macros through the pseudomacro facility, which consists of a set of predefined CLI instructions with a special syntax. The CLI supports these pseudomacros: !EQUAL, !NEQUAL, !END, !ELSE, !ERROR, !READ, and !RETURN.

Textual substitution enables you to enter a CLI command or a macro name as an argument to another command or macro. The command line will then function according to the current value of the substituted command or macro. Textual substitution is very useful both in interactive work with the CLI, and in macros.

The remainder of this chapter explains how you can create and invoke macros, and how you use pseudomacros and textual substitutions.

Macro Names

Any legal filename can be a legal macro name. The name you assign to a new macro is used to invoke that macro. Although it is not required, macro names are conventionally identified with the .CLI extension.

Invoking a Macro

To invoke or execute a macro, you enter a macro call. The macro call tells the CLI to find the macro file, substitute its contents for the macro call, and then process the contents in the order they appear, as illustrated in Figure 6.1.

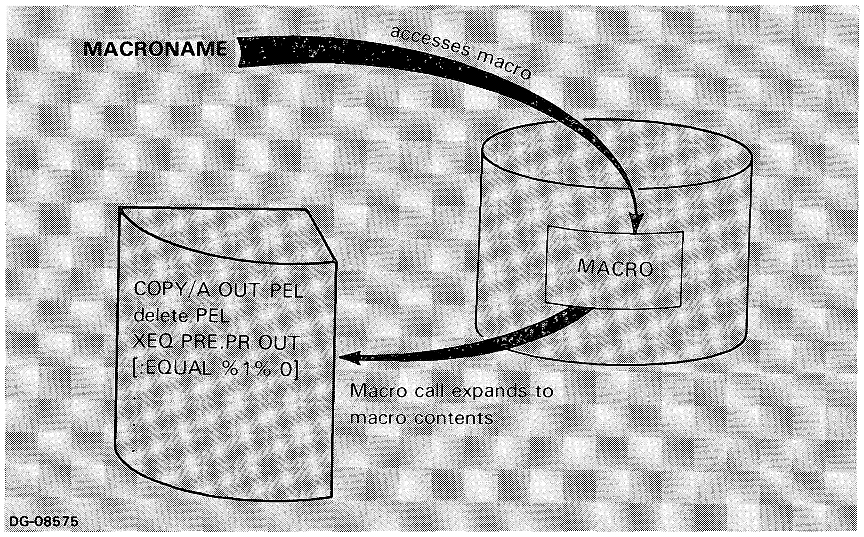


Figure 6.1 Invoking a macro

There are two ways to invoke a macro: explicitly or implicitly. Both are related to the order in which the CLI interprets the first entry in the command line, as shown in Figure 6.2.

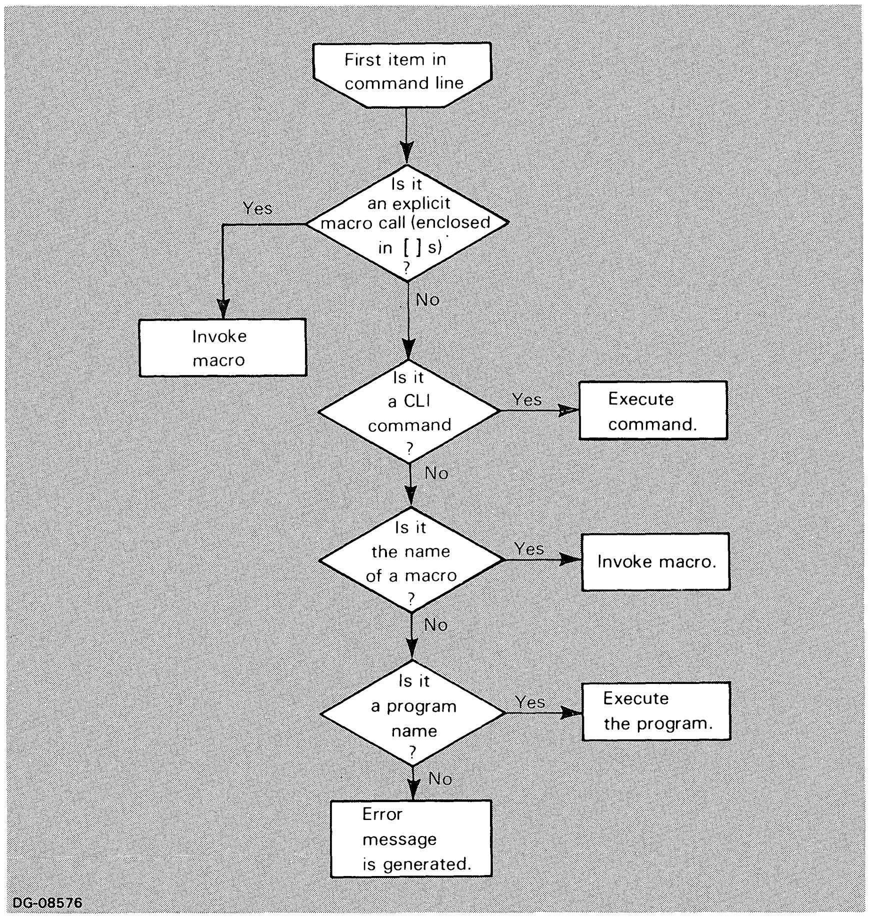


Figure 6.2 CLI evaluation of first item in command line

Explicit invocation indicates to the CLI that you are executing a macro, not a CLI command. To do this, enclose the macro name along with the macro's actual arguments in square brackets. That is,

[macroname argument-list] ↵

NOTE: *If the macro name includes the .CLI extension, using the extension is optional when invoking the macro.*

To explicitly invoke the TEST.CLI macro with argument list QUEUE, enter

```
) [TEST.CLI QUEUE] ↵
```
or
```
) [TEST QUEUE] ↵
```

If the macro call is the first entry in the command line, you may invoke the macro implicitly by typing the macro name without brackets; e.g.,

```
) TEST.CLI QUEUE ↵
```
or
```
) TEST QUEUE ↵
```

would invoke TEST.CLI implicitly. The CLI does not recognize this immediately as a macro call. Hence, the CLI first attempts to process this as a command, and failing, it looks for a macro by that name.

NOTE: *If you have a macro with the same name as a CLI command, you must execute the macro explicity.*

Besides invoking a macro by itself, you can also invoke it as an argument to another macro or CLI command. If the macro contains an argument sequence instead of a command sequence, it is invoked as an include file. In the case that the macro contains a command sequence, the macro is invoked as a textual substitution (discussed later in this chapter).

Using Include Files

An include file is a macro that contains a list of arguments, instead of a sequence of commands. This type of macro is always invoked as an argument to a command or macro. Invoke the macro by entering its name in square brackets.

For example, suppose you frequently specify filenames BEFORE, CHECK1, CHECK2, and AFTER as arguments to various programs you execute. If you write the filenames into the macro FILES, then you can specify the filename list by simply entering the macro name in brackets. To execute program THERM_TEST.PR and specify these files as arguments to the program, enter

```
) XEQ THERM_TEST.PR [FILES] ↲
```

Creating Macros

Macro files are created in the same way as text files — either with the CREATE command or with a text editor, such as SPEED.

Usually, a macro executes a sequence of CLI commands. You can include any CLI commands you want.

Suppose you want to write a macro to execute program DESIGN.PR and then display file REPORT. To create this macro with the CREATE command and name it SHOW.CLI, you would enter the following lines. Remember, CREATE/I prompts for console input with)).

```
) CREATE/I SHOW.CLI ↲
))XEQ DESIGN.PR ↲
))TYPE REPORT ↲
))) ↲
```

When you invoke SHOW.CLI, it will execute the commands you entered in the order they appear.

In-Line Comments

To include comments the CLI discards during execution of the macro, enter the symbol | and follow it with the comments. The CLI will not process any characters following the | to the end of the command line. For example, the macro FLOW.CLI contains these lines:

```
PUSH
DIRECTORY :THERMO
XEQ/S FLOW.PR PRIME | ANALYSE FLOW RATE PRIMARY SYSTEM
POP
```

NOTE: *The symbol | is shown as a broken* vertical bar on some terminals.

Whether or not you include comments, processing of the macro is unchanged.

Dummy Arguments

You can create a macro that will accept different arguments whenever you invoke it. This macro will contain *dummy argument* references.

Dummy arguments are variables for which actual arguments can be substituted. The actual argument is the string or number on which the command operates. In the command

```
DELETE FILEA
```

FILEA is the actual argument. However, in the command

```
DELETE %1%
```

The %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. Percent signs always enclose dummy arguments. This is how the CLI distinguishes them from actual arguments.

Dummy arguments are only used in macros. A dummy argument can represent one actual argument, a range of actual arguments, or a switch. Table 6.2 summarizes the format of the three types of arguments, and they are described below.

Single Dummy Arguments

The simplest type of dummy argument contains a single number enclosed by percent signs. The number represents the numerical position of the actual argument 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%. Hence 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 enter fewer arguments than dummy references, the CLI replaces the unmatched dummy arguments with the null string. However, if you enter more arguments than dummy arguments, the excess actual arguments are ignored.

Range Argument References

A range argument reference, as its name suggests, enables you to use a range of arguments in a macro. A range reference is entered as *%m-n%*. The CLI replaces this reference with arguments *m* through *n*, separated by commas.

NOTE: *If m* is omitted, 1 is assumed. If *n* is omitted, 32,767 is assumed.

Table 6.1 lists the various ways you can enter a range reference.

Dummy Argument Format	Default Value
%*m*-*n*%	Expands to every argument from the *m*th to the *n*th
%-*n*%	Expands to every argument from the first to the *n*th
%*m*-%	Expands to every argument from the *m*th to the last
%-%	Expands to every argument from the first to the last

Table 6.1 Range argument default values

Table 6.2 shows examples of argument range references.

Macro Invocation	
) [MAP COORD_X COORD_Y COORD_Z DIMENSION/VECTOR SCALAR] ↲	
Dummy Reference	**Reference Expands to:**
%1-5%	COORD_X,COORD_Y,COORD_Z,DIMENSION/VECTOR,SCALAR
%4-%	DIMENSION/VECTOR,SCALAR
%-%	COORD_X,COORD_Y,COORD_Z,DIMENSION/VECTOR,SCALAR
%0-%	MAP,COORD_X,COORD_Y,COORD_Z,DIMENSION/VECTOR,SCALAR
) FOO FILESTATUS/ASSORTMENT/SORT/TYPE=DIR ↲	
%1/SORT%	/SORT
%1\SORT%	/ASSORTMENT/TYPE=DIR
%1\SORT\ASSORTMENT%	/TYPE=DIR
%1/%	/ASSORTMENT/SORT/TYPE=DIR
%1\%	FILESTATUS
%1/TYPE%	TYPE=DIR
%1/TYPE=%	DIR
%1/LENGTH%	<null string>

Table 6.2 Dummy argument and switch reference examples

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 including a switch is:

%dummy-arg/switchname%

The format for excluding a switch is:

%dummy-arg\switchname%

NOTE: *If you do not specify* dummy-arg, *argument 0 is assumed.*

To exclude more than one switch, use a \ before each one.

If you type a slash without a switchname, then the CLI assumes that you are referring to all switches for the particular argument. If you type a backslash without a switchname, the CLI returns the specified argument without any of its switches.

If the switch you specify in the dummy argument takes a value (i.e., is a keyword switch), its value is automatically included in the expansion.

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

%dummy-arg/switchname=%

For example, using the second command line example in Table 6.2, %1/TYPE=% would be replaced with DIR.

If the dummy reference refers to a switch that is not included in the macro call, the CLI replaces the reference with the null string.

Table 6.2 shows examples of switch dummy argument references. Table 6.3 summarizes the various formats and functions for dummy arguments.

Dummy Argument	Expansion
%0%	Expands to macro name (with all its switches)
%*n*%	Expands to the *n*th argument (with all its switches)
%*m*-*n*%	Expands to every argument from the *m*th to the *n*th
%*n*/%	Expands to the *n*th argument's switches
%*n*\%	Expands to the *n*th argument without its switches
%*n*/s%	Expands to the *n*th 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 *n*th argument's /s=switch
%*n*\s%	Expands to all of the *n*th argument's switches except /s
	NOTE: You may include more than one slash or backslash in a dummy argument. However, you may not include both a slash and a backslash in the same dummy argument.

Table 6.3 Dummy argument formats

Using the /M Switch in a Macro

The CREATE and EXECUTE (or XEQ) commands accept the /M switch, which allows these commands to take input from subsequent lines of the macro. In each case, the command accepts input until it reaches a line that contains a right parenthesis and no other characters (except an optional New-line terminator). The CLI passes all of the input lines verbatim to the file or program, with the following exceptions: it substitutes values for all dummy arguments, passes literals intact but without their double quote delimiters, and combines continuation lines, eliminating any ampersands (&). Once it encounters the), the macro continues processing.

For instance, you may wish to use a macro to create a file containing a reminder for later retrieval.

```
PUSH
DIRECTORY :CALENDAR
CREATE/M %1%
Why quota not filled by 25th?
)
POP
```

This macro makes :CALENDAR the working directory, writes the indicated text string into a file specified by the user of the macro, and moves back to the original working directory.

Similarly, consider the following macro:

```
PUSH
MEASURE ON
XEQ/M GENERATOR.PR
A B C D E
DATA_LIST
7243
)
POP
```

The macro moves up one level in the CLI environment, and turns on the elapsed-time reporting mode. Then it executes the program GENERATOR.PR, passing it three lines of data. The POP command restores the CLI environment to its original level.

Displaying a Text String

If you want to display some text in your macro's output, use the WRITE command. The format for WRITE is

)WRITE *[argument-1 ...argument-n]* ⤶

This command writes the arguments you specify to the output device exactly as you enter them. For instance,

```
) WRITE HERE I AM ⤶
```

displays

HERE I AM

If you include the following lines in a macro,

```
WRITE THE CURRENT ENVIRONMENT IS
CURRENT
WRITE THE PREVIOUS ENVIRONMENT IS
PREVIOUS
```

the CLI displays THE CURRENT ENVIRONMENT IS before processing the CURRENT command, and then displays THE PREVIOUS ENVIRONMENT IS before executing the PREVIOUS command.

The macro facility, as described so far in this chapter, is a versatile tool for saving you time and effort. You can enhance this facility even further by incorporating pseudomacros and textual substitutions. Both are similar in format but different in function.

Pseudomacros

A pseudomacro is a predefined CLI instruction with a special syntax that can be used to add further dimension to your macro. It enables you to execute a single instruction or a sequence of instructions. The general format for entering a pseudomacro is

[pseudomacro arguments]

The pseudomacro and its argument (if any) must be enclosed in square brackets. Table 6.4 lists and briefly describes the pseudomacros supported by the CLI pseudomacro facility. Each is described in more detail below.

Pseudomacro	Action
!ELSE	Execute an alternate sequence of commands when !EQUAL or !NEQUAL is false
!END	Complete an !EQUAL or !NEQUAL block
!EQUAL	Execute a block of commands when equality condition is true
!NEQUAL	Execute block of commands when inequality condition is true
!READ	Display message prompt and wait for keyboard input
!RETURN	Terminate macro and return argument string to macro call

Table 6.4 CLI pseudomacros

NOTE: *A pseudomacro can be used only in a macro or as an argument to a CLI command.*

Conditional Pseudomacros

As indicated in Table 6.4, !EQUAL and !NEQUAL are conditional pseudomacros. These determine which arguments are passed to a command and which commands execute, depending on whether or not the specified condition is satisfied.

If the condition is true, the CLI continues processing according to the prescribed sequence that follows. Each conditional pseudomacro sequence is completed with the !END pseudomacro.

If the condition is false, the CLI skips the entire sequence, unless you have included the !ELSE pseudomacro with an alternative sequence.

The different paths the CLI can follow when it encounters a conditional pseudomacro are illustrated in Figure 6.3.

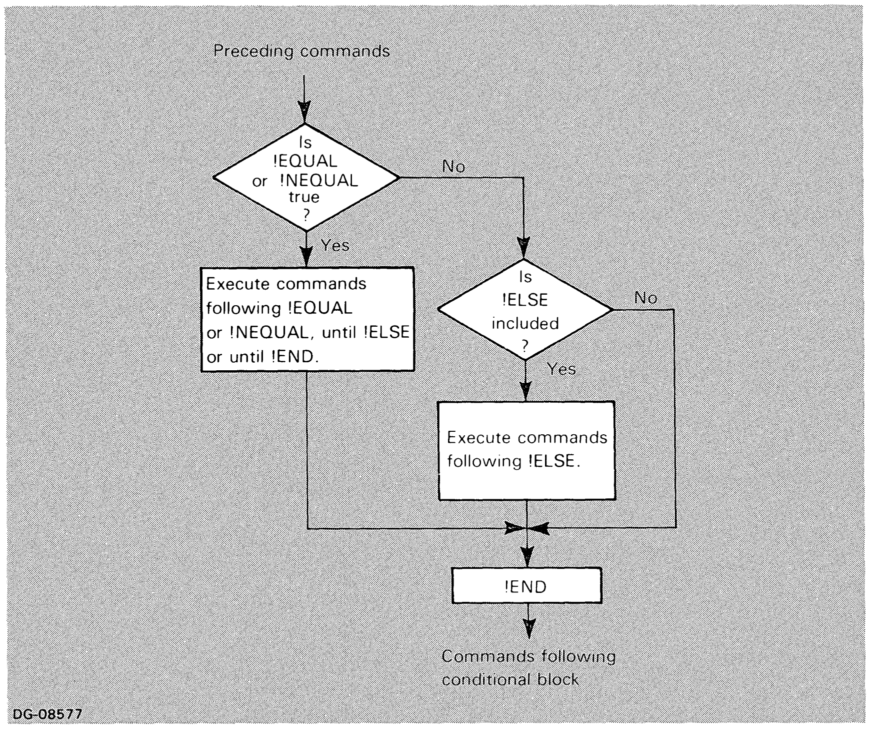

Preceding commands
Is !EQUAL or !NEQUAL true ?
No
Yes
Execute commands following !EQUAL or !NEQUAL, until !ELSE or until !END.
Is !ELSE included ?
No
Yes
Execute commands following !ELSE.
!END
Commands following conditional block
DG-08577

Figure 6.3 Execution of a conditional pseudomacro path

!EQUAL

!EQUAL compares two arguments and continues processing if the two arguments are identical.

[!EQUAL *argument-1,argument-2*]

You must enter two arguments for !EQUAL to compare. Each is compared character by character. You should use commas to separate the arguments, or spaces if neither argument is (or might be) null. If either argument is the null string, use commas to indicate this fact.

If the arguments evaluate as identical strings, the condition is true and the CLI processes the rest of the sequence that follows. If the two arguments are dissimilar, the !EQUAL pseudomacro returns a false, and the CLI skips to the block's !END statement (or !ELSE block, if included), and then continues processing. If an !ELSE block has been included, it is processed only when the !EQUAL condition is false.

```
[!EQUAL 99,%1%]
BYE
[!END]
```

In the example above, the CLI currently executing shuts down when the first argument in the macro call is equal to 99. If %1% is not 99, the CLI continues processing after !END.

```
XEQ LOGIC.PR
[!EQUAL,,%1%]
XEQ REPORT.PR
[!END]
```

Here !EQUAL compares the null string to the first argument in the macro call. If there are no arguments in the macro call, the condition is true, and the CLI executes REPORT.PR.

```
[!EQUAL TOGGLE,%1%]
  WRITE LIGHTS ON
[!END]
[!EQUAL,%1%,]
  WRITE LIGHTS OFF
[!END]
```

This macro displays the string LIGHTS OFF if argument 1 in the macro call is TOGGGLE; otherwise (!ELSE) it displays LIGHTS OFF.

!NEQUAL

!NEQUAL compares two arguments and continues processing if the two arguments are NOT identical.

[!NEQUAL *argument-1 argument-2*]

As with the !EQUAL block, the !NEQUAL block must be completed with !END.

If the arguments are dissimilar, the condition is true, and the CLI executes the commands you include in the block. If the arguments are identical, the condition is not true. Hence, the CLI skips the rest of the block and begins processing after !END (or after !ELSE, if included in the block). For example,

```
[!NEQUAL 9,%1%]
XEQ COUNT.PR %1%
[!END]
```

compares the text string 9 to the first argument in the macro call. As long as they are not equivalent, the CLI passes the first argument to program COUNT.PR and executes the program. If they are equal, the CLI skips the rest of the block and continues processing after [!END].

Using Conditionals in Recursive Macros

You can use !EQUAL or !NEQUAL to control recursive macros.

For example, the conditional block in the macro might contain a macro call to re-invoke the macro. When the condition is true, the macro is invoked again; when it is false, the block is skipped.

The sample macro SYS_REP.CLI is recursive. It contains these lines:

```
XEQ REPORT.PR %1%
  [!NEQUAL ,, %2%]
SYS_REP.CLI %2-%
  [!END]
```

The !NEQUAL block contains the macro call that invokes SYS_REP.CLI again. As long as the second argument in the original macro call is not null, the macro will be re-invoked.

Figure 6.4 illustrates what happens if you invoke SYS_REP.CLI with arguments SYS1, SYS2, and SYS3.

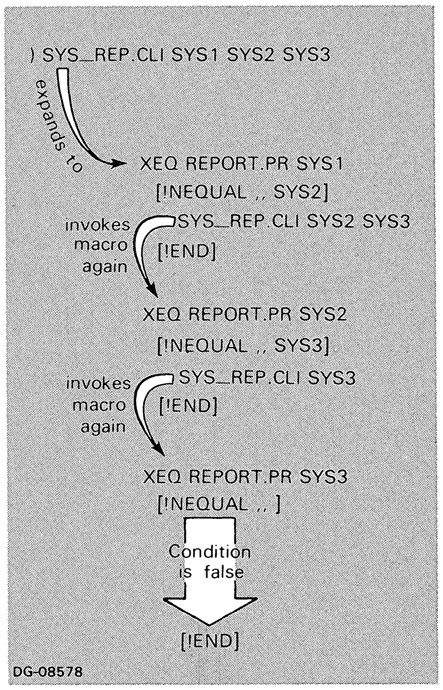


Figure 6.4 Processing of a recursive macro

!ELSE

This pseudomacro enables you to include an alternative sequence of command lines in an !EQUAL or !NEQUAL block to be executed when the condition is false.

[!ELSE]

Place !ELSE and its following command sequence between the conditional statement (!EQUAL or !NEQUAL) and the block's !END statement, i.e.,

[!EQUAL ...]
command sequence
[!ELSE]
alternative command sequence
[!END]

If the conditional statement is true, the CLI ignores the !ELSE sequence and continues processing after !END. However, when the condition is false, the CLI ignores the input lines between the conditional statement and !ELSE, continuing to process at the !ELSE line.

```
[!NEQUAL OIL,%1%]
  XEQ ALTERNATE.PR %1%
[!ELSE]
  XEQ OIL.PR
[!END]
```

This macro compares the text string OIL to the first argument in the macro call. If they are not equivalent, the CLI executes ALTERNATE.PR and skips the !ELSE sequence. If !NEQUAL is false, the CLI skips to the !ELSE block and continues processing, thereby executing OIL.PR.

NOTE: *If you nest conditional blocks (discussed later), the !ELSE block is associated with the innermost open conditional block.*

!END

The !END pseudomacro marks the last line of any !EQUAL or !NEQUAL block (described further in this chapter). Its format is:

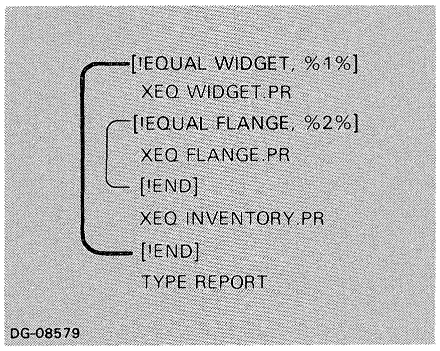
[!END]

Nested Conditionals

You can nest conditional pseudomacros if you wish. !END completes the innermost open block, as illustrated by Figure 6.5.

This command sequence contains two !EQUAL blocks terminated by !END statements.

The second !EQUAL block is nested in the first. Notice that the *first* !END pseudomacro completes the second block. The final !END then completes the outer !EQUAL block. How this macro actually executes depends on whether or not the conditions are satisfied, as illustrated in Figure 6.6.



```
[!EQUAL WIDGET, %1%]
XEQ WIDGET.PR
[!EQUAL FLANGE, %2%]
XEQ FLANGE.PR
[!END]
XEQ INVENTORY.PR
[!END]
TYPE REPORT
```

Figure 6.5 Nested conditionals

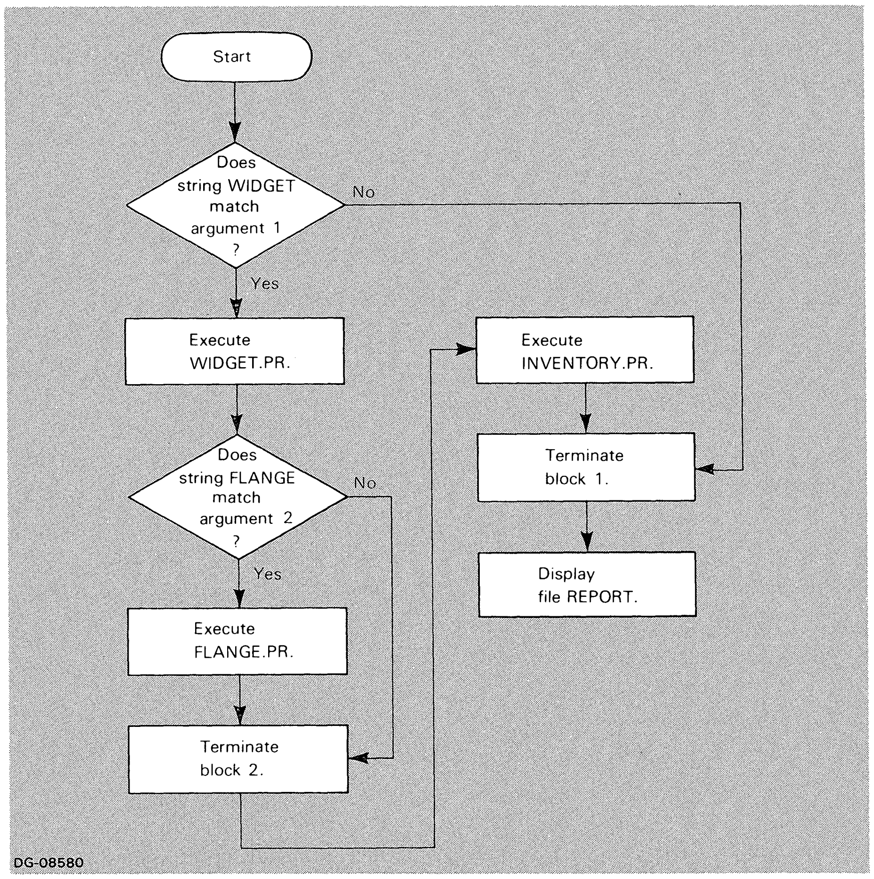


Figure 6.6 Flow through example of nested conditional blocks

If you include the !ELSE pseudomacro, it is associated with the innermost open block.

Unbalanced Conditionals

If you attempt to execute a macro that does not contain enough [!END]s to complete all the open conditional blocks, the CLI will issue an error message and terminate the macro(s) it is executing.

Other Pseudomacros

The !READ Pseudomacro

The !READ pseudomacro displays a message as a prompt and awaits further input from the keyboard. Its format is

[!READ *message*]

Enter a *message* that will indicate the kind of response you must enter when the !READ prompts you for input. Your response can be a maximum of 127 characters, and must be completed with a New-line or Carriage Return.

!READ must always be entered as an argument to another CLI command. It uses the response to the prompt as an argument to the rest of the command line.

```
PUSH
DIRECTORY @DPDO:SYSTEMS
XEQ RANGE.PR [!READ NUMBER OF DIMENSIONS? ENTER 1,2,OR 3]
POP
```

In this example, !READ is an argument to the XEQ command. Whatever is entered as a response to !READ will be passed to program RANGE.PR. When the !READ message is displayed, you will know your choices of responses are 1, 2, or 3.

The next macro displays information about French or Italian cuisine, depending on your response to WHICH DO YOU PREFER? displayed by !READ. If you type FRENCH, the CLI executes the !EQUAL block, displaying the file FOOD.FRENCH. However, if you type ITALIAN, (or *anything* other than FRENCH) the CLI skips to the !ELSE pseudomacro and continues processing there, displaying the file FOOD.ITALIAN instead.

```
WRITE THE RESTAURATEUR GOES FRENCH OR ITALIAN
[!EQUAL FRENCH,[!READ WHICH DO YOU PREFER? (FRENCH/ITALIAN) ]]
  TY FOOD.FRENCH
  [!ELSE]
   TY FOOD.ITALIAN
[!END]
```

The !RETURN Pseudomacro

!RETURN terminates the macro currently processing. If the macro was invoked using textual substitution, !RETURN passes any included arguments to the command line that called the macro; if you do not specify an argument string, !RETURN passes the null string.

[!RETURN *[argument-string]*]

The following is a simple example of how !RETURN functions.

```
) [!TEST.CLI] ↲

(TEST.CLI contains a !RETURN)
      .
      .
      .
  [!RETURN !FILESTATUS]

DIRECTORY @DPDO:PRESSURE

MASS   TEMPERATURE  VOLUME

)
```

When the CLI processes the !RETURN statement, the macro ends. Since the macro was invoked using a textual substitution, the argument string FILESTATUS is substituted for the macro call. The CLI then executes this string as a command.

The next macro, TIMER.CLI, includes a !RETURN that passes the null string to the command line invoking the macro.

```
) TIMER.CLI ⤶
```

(The macro contains these lines:)

```
XEQ TIME.PR,%1%
 [!EQUAL 0,%1%]
   [!RETURN]
 [!END]
DELETE TIME__TEMP
)
```

If the !EQUAL condition is true, the !RETURN statement is executed, thereby terminating the macro and replacing the original macro call with the null string. If !EQUAL is false, the CLI skips to after !END and deletes TIME_TEMP.

Textual Substitution

You can substitute output from a CLI command or macro as an argument to another CLI command or macro by using "textual substitution".

You may want to substitute the current value for a command, such as the system date, or the result of an arithmetic operation (arithmetic functions are described in Chapter 7). To do this, enter the command as:

[*!command*]

where *command* is any CLI command or macro. This format is similar to that for pseudomacros.

If no value results from execution of the command you enter, the CLI returns a null string.

Suppose your macro contains

```
[!NEQUAL 0 [!LEVEL]]
PREVIOUS
[!END]
```

The current environment level number is substituted for [!LEVEL] and is consequently passed as an argument to the !NEQUAL pseudomacro. In this case, when LEVEL is not 0, the CLI displays the previous environment level.

You can include arguments and switches with the substituted command; however, the switches can only display information, not change it. For example, the command

```
)CHARACTERISTICS/I ↵
```

resets the console characteristics to their initial values. However,

```
)WRITE [!CHARACTERISTICS/I] ↵
```

will simply write the values of the initial characteristics on the console screen—the characteristics themselves will remain unchanged. Furthermore, the command WRITE [!CHARACTERISTICS/OFF/BIN] will not work at all, since the textual substitution attempts to reset a value.

A textual substitution can be nested within another, as in this command line:

```
) ADD %4% [!ADD %3% [!ADD %1% %2%]] ↵
```

where the sum of arguments 1 and 2 is added to argument 3, and then that sum is added to argument 4 to calculate the final sum.

To what level you can nest textual substitutions depends on the current memory constraints. However, an error in a nested substitution will cause an error in the outermost substitution level.

Where the command you use as a textual substitution normally returns a multi-part output, the CLI substitutes a string consisting of the multiple terms separated by commas only. Any headings, embedded New-lines, trailing or leading New-lines, spaces, and tabs are removed before substitution. For instance, the FILESTATUS command returns all the filenames in the current working directory. Suppose the current working directory is @DPD0:MATERIAL, and it contains files XYLON, MYLON, and VILOX. Entering

```
) FILESTATUS ↲
```

displays

```
DIRECTORY @DPDO:MATERIAL

XYLON    MYLON    VILOX
```

Including FILESTATUS as a textual substitution in a command line will substitute this information as shown in Figure 6.7. (The heading showing the working directory is removed before substitution.)

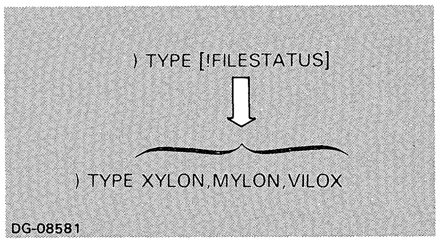


Figure 6.7 Textual substitution example

The CLI then processes the TYPE command with these arguments.

Textual Substitutions in Macros

The textual substitution facility is equally useful in macros. Consider the following macro, STRUCTURE.CLI.

```
[!NEQUAL,%1%,0]
XEQ CRYSTAL.PR TEST%1%
STRUCTURE [!SUBTRACT %1% 1]
[!END]
```

The SUBTRACT command is used as a textual substitution in this macro. The value calculated by subtracting 1 from the first argument in the macro call becomes an argument to STRUCTURE, which invokes the macro again. (This is a recursive macro, invoking itself repeatedly until the !NEQUAL condition is false.)

In the following example, both the macro SCHEDULE.CLI and the command STRING are entered as textual substitutions.

```
) FILESTATUS/AS [!SCHEDULE.CLI] ↲
```

(SCHEDULE.CLI contains these lines:)

```
XEQ/S PLAN.PR
  [!EQUAL,,[!STRING/S=1]]
  COPY/A GANNT CURRENT
  DELETE CURRENT

  [!END]
[!RETURN [!STRING/S=1]]
```

The termination message from the execution of program PLAN.PR is written to String Buffer 1. When STRING is executed, its value (the contents of String Buffer 1) is compared to the null string. If !EQUAL is true, the rest of the block is processed; otherwise, the CLI skips it. In either case, the !RETURN passes the contents of String Buffer 1 as an argument to the FILESTATUS command.

Exploding a Text String

The EXPLODE command allows you to separate any text string into its individual characters. You can then pass whichever characters you want as arguments to another CLI command or macro. Enter this command as

```
) EXPLODE string ↲
```

EXPLODE converts all spaces and tabs in the text string to commas, and then places a comma between every two characters (other than commas) in the new string. For instance,

```
) EXPLODE NEGZS# 0,1,SEZ ↲
N,E,G,Z,S,#,,,0,,,1,,,S,E,Z
```

The spaces in this text string were converted to commas before the characters were separated.

In the next example, the system date is used as a textual substitution to the EXPLODE command. The date contains no commas or spaces.

```
) DATE ↲
30-JAN-81
) EXPLODE [!DATE] ↲
3,0,-,J,A,N,-,8,1
```

Figure 6.8 illustrates how you might use EXPLODE and textual substitution in a macro to separate a string into single characters and then select a few of the characters as arguments to another macro or command.

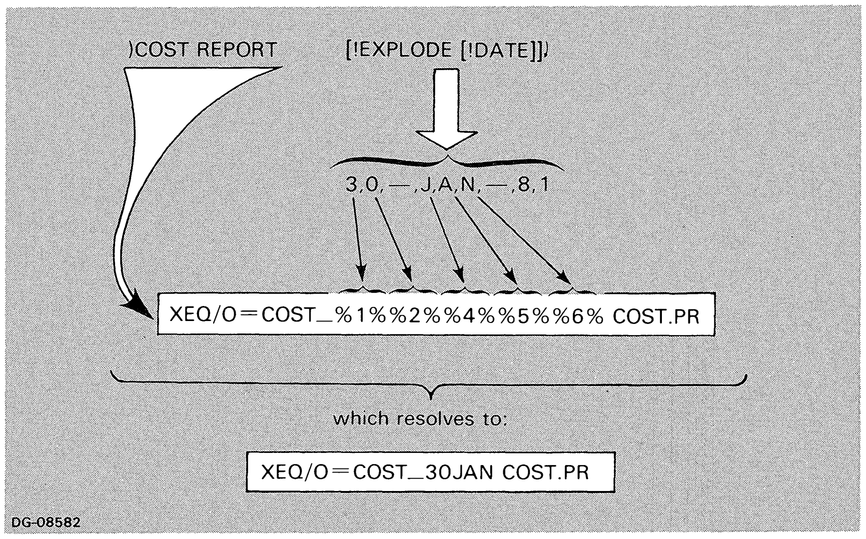


Figure 6.8 Exploding a text string to single character arguments

The example in this figure explodes the system date and passes some of the resulting characters to another macro, COST_REPORT. Notice how only certain characters are actually substituted in the dummy references; the others are ignored.

Both the DATE and EXPLODE commands are entered as textual substitutions in this example.

Removing Final Extensions

Another command that may be useful in a macro is NAME. This command lists the arguments you specify without their final extensions.

) NAME *argument-1 [,...,argument-n]* ↲

If an argument contains more than one period, only the last extension is removed.

```
) NAME  LOOMA.FR  FOO.SR.BU ↲
LOOMA  FOO.SR
```

This example displays two arguments without their final extensions. Notice that FOO.SR.BU is displayed as FOO.SR. Only the final extension is dropped.

You might want to use NAME as a textual substitution. For example,

```
) WRITE [!NAME [!FILESTATUS +.PAS]] ↲
```

This will display the filenames of all the PASCAL source files in the current working directory, without the .PAS extension.

Debugging Macros

The CLI Trace mode is useful for debugging macros. When trace is on, it displays each command line as it appears to the CLI before the CLI executes it. The TRACE command turns Trace mode on or off.

) TRACE {ON | OFF} ↲

By default, Trace mode is off. ON turns it on, and OFF turns it off. If you enter TRACE without an argument, the CLI displays the current trace mode setting.

When Trace mode is on, CLI commands are preceded with ***, textual substitutions and pseudomacros with !!!, and macro calls with ###.

The macro INCREMENT.CLI below

```
 [!EQUAL,%1%,2]
   WRITE DONE
 [!ELSE]
   WRITE %1%
   INCREMENT [!UADD,1,%1%]
 [!END]
```

produces the following trace output:

```
INCREMENT 0
### [INCREMENT,0]
!!! [!EQUAL,0,2]
!!! [!ELSE]
*** WRITE,0
0
!!! [!UADD,1,0]
### [INCREMENT,1]
!!! [!EQUAL,1,2]
!!! [!ELSE]
*** WRITE,1
1
!!! [!UADD,1,1]
### [INCREMENT,2]
!!! [!EQUAL,2,2]
*** WRITE,DONE
DONE
!!! [!ELSE]
!!! [!END]
```

Although the CLI displays an explicit error message when an error occurs, a complex macro with repeated commands may make it difficult to find the line causing an error. Trace mode, however, helps you find the line more easily. It is also particularly helpful in locating logic errors.

Timing Command Execution

The MEASURE command allows you to find out how long every CLI command takes to execute.

) MEASURE {ON | OFF} ↵

The following macro uses MEASURE to record the phases of its own execution.

```
PUSH
PROMPT POP
MEASURE/L=MACROTIME ON
    .
    .
    .
[other commands]
```

Since MEASURE sets part of the CLI environment, the PUSH and PROMPT POP commands ensure that the mode will be on only during the execution of the macros.

The LOGON.CLI Macro

The LOGON.CLI macro is automatically executed whenever the root CLI is invoked. It is executed before the first CLI prompt appears at your console.

This macro is a useful place to put messages and commands you wish to have executed each time you start up the CLI. For instance, you may want to reset your searchlist and working directory from their initial settings.

Arithmetic & Conversion Functions

MP/AOS CLI includes several commands to perform arithmetic and conversion functions. It is possible to add, subtract, multiply, and divide, or perform a modulus operation. You can also convert octal numbers to their corresponding ASCII characters, octal numbers to decimal numbers, or decimal numbers to octal numbers.

The commands you can use to perform these arithmetic and conversion functions are listed in Table 7.1.

Command	Action
ASCII	Convert an octal number to corresponding ASCII character
DECIMAL	Convert an octal number to corresponding decimal number
ERCODE	Display the five-digit octal error code resulting from the last CLI command
MESSAGE	Display text message corresponding to a specified error code
OCTAL	Convert a decimal number to corresponding octal number
UADD	Calculate the sum of two decimal integers
UDIVIDE	Calculate the integer quotient of two integers
UMODULO	Perform a modulus operation
UMULTIPLY	Calculate the product of two decimal integers
USUBTRACT	Calculate the difference between two decimal integers

Table 7.1 CLI arithmetic and conversion commands

This chapter describes each of these commands.

Arithmetic Commands

The CLI arithmetic commands are UADD, USUBTRACT, UMULTIPLY, UDIVIDE, and UMODULO. These enable you to perform basic arithmetic functions. Each is described below.

Adding Two Numbers

UADD computes and returns the sum of two unsigned, decimal integers. Enter it as

UADD *integer1 integer2* ↲

Both numbers you add must be unsigned, decimal integers in the range 0 to 4,294,967,295. For instance, calculate the sum of 255 and 690.

```
) UADD 255 690 ↲
945
```

If you add to a number at the top of the range, the result "wraps around" to the bottom of the range. For example,

```
) UADD   4294967295   3 ↲
2
```

You may want to perform an arithmetic operation in a macro. The following example uses UADD as a command and as a textual substitution to calculate the sum of five integers.

```
) SUM_5 290 35 79 101 215 ↓
```

(SUM_5.CLI is a macro that contains these lines)

```
  WRITE THE SUM IS
  UADD %1% [!UADD %2% [!UADD %3% [!UADD %4% [!UADD %5%]]]]
```

```
THE SUM IS
720
)
```

Subtracting Two Numbers

UBSUBTRACT subtracts one unsigned, decimal integer *(integer2)* from another *(integer1)* and returns the result.

) USUBTRACT *integer1 integer2* ↓

Both integers must be in the range 0 to 4,294,967,295. The result will be an unsigned, decimal integer in the same range.

```
) USUBTRACT 350 119 ↓
231
```

Subtracts 119 from 350 and displays the result, 231.

If the subtraction produces a negative number, the number "wraps around" the number range. That is, counting backward from zero, the next "lowest" number is 4,294,967,295. Thus, a negative number is shown as a very large unsigned number. For example:

```
) USUBTRACT 3 12 ↓
4294967287
```

Multiplying Two Numbers

The UMULTIPLY command computes and then returns the product of two unsigned decimal integers.

) UMULTIPLY *integer1 integer2* ↓

Both arguments are unsigned decimal integers. *Integer1* must be in the range 0 to 4,294,967,295; *integer2* must be in the range 0 to 65,535.

```
) UMULTIPLY 300 12 ↓
3600
```

Multiplies 300 by 12 and displays the product, 3600.

Dividing Two Numbers

The UDIVIDE command returns the integer quotient that results from dividing one unsigned integer by another.

```
) UDIVIDE dividend divisor ↵
```

The CLI divides the *dividend* by the *divisor*. The *dividend* must be an unsigned integer in the range 0 to 4,294,967,295, whereas the *divisor* must be an unsigned integer in the range 1 to 65,535. Remainders are discarded, hence the result is an integer.

```
) UDIVIDE 255 15 ↵
17
```

This divides 255 by 15, and lists the resulting integer quotient.

```
) UDIVIDE 1 5 ↵
0
```

UDIVIDE calculates the integer quotient for 1 divided by 5. The remainder is discarded.

Performing Modulo Operations

The UMODULO command performs a modulo operation.

```
) UMODULO integer modulus ↵
```

Integer is in the range 0 to 4,294,967,295. *Modulus* can range from 0 to 65,535.

```
) UMOD 12 8 ↵
4
```

Computes 12 modulo 8. The result is 4.

Conversion Commands

The conversion commands convert numbers to their ASCII, decimal, or octal equivalents; or they supply error code and message information. The remainder of this chapter describes the ASCII, DECIMAL, OCTAL, ERCODE, and MESSAGE commands.

Converting to ASCII

ASCII converts octal numbers to their corresponding ASCII values and returns the results to the command line.

```
) ASCII octal-number-1 [...octal-number-n] ↵
```

The specified numbers must be positive integers in the range 0 to 377.

```
) ASCII 50 51
()
)
```

Converts the octal numbers to their corresponding ASCII values: right and left parentheses.

You may wish to use ASCII (or any of the other conversion commands) in a macro. The next example converts the octal numbers that are returned to String Buffer 1 by program TRANSLATE.PR to their corresponding ASCII characters. These characters are then displayed by the WRITE command.

```
  XEQ/S TRANSLATE.PR %1%
  WRITE [!ASCII [!STRING]]
```

Converting to Decimal

DECIMAL converts an octal number to its corresponding decimal value and returns the result to the command line.

) DECIMAL *octal-number* ⤸

The octal number you specify must be an unsigned integer in the range 0 to 37,777,777,777.

```
) DECIMAL 100 ⤸
64
```

Expands to the decimal equivalent of the octal number 100.

Converting to Octal

OCTAL converts an unsigned decimal number to its octal equivalent and returns the result to the command line.

) OCTAL *decimal-number* ⤸

You can convert any decimal integer in the range 0 to 4,294,967,295.

```
) OCTAL 10 ⤸
12
```

Converts 10 in decimal to octal and displays the result.

Retrieving a CLI Error Code

Every CLI command, whether it executes successfully or not, returns an error code. A successful command returns a zero. The ERCODE command returns the most recent error code as an octal number. If the appropriate exception condition has been set to IGNORE, use ERCODE to retrieve the error code.

```
) ERCODE ⤸
```

The value returned by ERCODE can be used either as a string or as an argument to an arithmetic operation.

```
) XEQ ERROR.PR [!UDIVIDE[!ERCODE]1024] ⤸
```

The above example executes ERROR.PR and passes to it the integer result of the error code divided by 1024.

Displaying Error Code Text

The command MESSAGE returns the text associated with system error code in either octal or decimal.

```
) MESSAGE error-code-1 [...error-code-n] ↲
```

This form of the command expects the error code number you supply to be in octal. Use the /D switch if the error code is in decimal.

The macro SPOT_IT.CLI contains the following lines

```
XEQ SPOT.PR
[!NEQUAL,0,[!ERCODE]]
WRITE/L=SPOT_LOG [!MESSAGE[!ERCODE]][!DATE][!TIME]
[!END]
```

These commands execute SPOT.PR. If that program returns a system error other than 0, the error text is written into the file SPOT_LOG, along with the date and time.

CLI Command Summary

8

This chapter summarizes the commands available in MP/AOS CLI according to related functions:

File Management (Chapter 3)
CLI Environment Management (Chapter 4)
System Parameter Management (Chapter 4)
Process Management (Chapter 5)
Program Management (Chapter 5)
CLI Pseudomacros (Chapter 6)
Arithmetic and Conversion Operations (Chapter 7)
Miscellaneous (various chapters)

For a detailed description of each command, refer to Chapter 9, "CLI Command Dictionary."

NOTE: *Command switches are omitted from this summary.*

File Management

The CLI enables you to create, delete, rename, and copy files. You can also display information about a file and its contents. Table 8.1 lists the commands you can use to manage your files.

Command	Action
ATTRIBUTES	Set or display a file's attributes
COPY	Copy one or more files into a destination file
CREATE	Create a file
DELETE	Delete one or more files
DIRECTORY	Display or modify the working directory
FILESTATUS	Display file information
PATHNAME	Display the fully-qualified pathname for the specified file
QPRINT	Place one or more file in the line printer output queue
RENAME	Change the name of a file
SEARCHLIST	Set or display the searchlist
TYPE	Type the contents of one or more files

Table 8.1 File management commands

CLI Environment Management

The commands listed in Table 8.2 manage the CLI environment. In addition, the DIRECTORY and SEARCHLIST file management commands also affect the CLI environment.

Command	Action
CHARACTERISTICS	Set or display the features of system character device
CLASS1	Set or display the CLI response to errors caused by commands affecting the CLI environment
CLASS2	Set or display the CLI response to errors caused by commands not affecting the CLI environment
CURRENT	Display the current CLI environment values
LEVEL	Display the current CLI environment level
MEASURE	Set or display the current setting for measuring command execution time
POP	Change the current CLI environment level to the next numerically lower level
PREVIOUS	Display the previous CLI environment level's values
PROMPT	Set or display the prompt command sequence
PUSH	Change the current environment level to the next numerically higher level
STRING	Set or display the string buffers
TRACE	Set or display the current setting for tracking command execution

Table 8.2 CLI environment commands

System Parameter Management

The commands listed in Table 8.3 enable you to manage certain aspects of the system configuration.

Command	Action
BOOT*	Shut down the current system
DATE	Set or display the system date
DISKSTATUS	Display disk information
DISMOUNT	Remove a disk device from the system
MOUNT	Make a disk file's structure available to the system
TIME	Set or display the system time

Table 8.3 Commands affecting system parameters

**Restricted to any swap level of initial process. Note that either BYE or CTRL-C CTRL-B, when issued from the CLI at Swap Level 1 in initial process, will also shut down the system.*

Process Management

Use the commands listed in Table 8.4 to create or terminate a process, or to stop or restart process execution. Also included in this table are commands you can use to display information about a process or to communicate with other processes in the system.

Command	Action
ACTIVE	Display current list of PID numbers in use
BLOCK	Stop a process
CONTROL	Send an interprocess communication message to a server process
PRIORITY	Return process priority of specified process
PROCESS	Create a process
RUNTIME	Report resource usage statistics
SYSTEM	Display the status of the system
TERMINATE	End a process
UNBLOCK	Restart a blocked process
WHO	Identify a process

Table 8.4 Process management commands

Program Management

The CLI enables you to execute or debug a program, display characteristics of a program, or change a program's revision number. The commands for managing your programs are listed in Table 8.5.

Command	Action
BYE	Terminate the CLI
CHAIN	Overwrite the CLI with new program
DEBUG	Invoke the Symbolic Debugger to debug a program
EXECUTE	Execute a program at next program level
INFORMATION	Display program characteristics
REVISION	Display and set the current program revision number
SWAPLEVEL	Display the program swap level of the current CLI
XEQ	Same as EXECUTE

Table 8.5 Program management commands

CLI Pseudomacros

Table 8.6 lists the pseudomacros you can use to enhance the macros you create.

Pseudomacro	Action
!ELSE	Execute an alternate sequence of commands when !EQUAL or !NEQUAL is false
!END	Complete an !EQUAL, or !NEQUAL block
!EQUAL	Execute a block of commands when equality condition is true
!NEQUAL	Execute block of commands when inequality condition is true
!READ	Display message prompt and wait for keyboard input
!RETURN	Terminate macro and return argument string to macro call

Table 8.6 CLI pseudomacros

Arithmetic & Conversion Operations

To perform arithmetic or conversion functions in the CLI, use the commands listed in Table 8.7.

Command	Action
ASCII	Convert an octal number to corresponding ASCII character
DECIMAL	Convert an octal number to corresponding decimal number
OCTAL	Convert a decimal number to corresponding octal number
UADD	Calculate the sum of two decimal integers
UDIVIDE	Calculate the integer quotient of two integers
UMODULO	Perform a modulus operation
UMULTIPLY	Calculate the product of two decimal integers
USUBTRACT	Calculate the difference between two decimal integers

Table 8.7 CLI arithmetic and conversion commands

Miscellaneous CLI Commands

The following commands perform functions not particularly associated with any of the aforementioned command categories.

Command	Action
ERCODE	Display the five-digit error code returned by the last CLI command
EXPLODE	Expand a text string into single characters
HELP	Display an explanation of a CLI command or general topic
MESSAGE	Display the text message corresponding to a specific error code
NAME	List the specified filenames without their final extensions
PAUSE	Stop the CLI
WRITE	Display the arguments you enter

Table 8.8 Miscellaneous CLI commands

CLI Command Dictionary

9

This chapter provides complete information on each MP/AOS CLI command, including format, switches, and examples. The commands are arranged alphabetically.

ACTIVE Display PIDs of Active Processes

)ACTIVE ↲

Returns the process identity numbers (PIDs) of currently executing (or blocked) processes.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) ACTIVE ↲
Active PIDs:
2
9
105
)
```

Shows the PID numbers of all processes currently running.

The macro ?.CLI contains the following command line:

```
WHO [!ACTIVE]
```

In this macro, the WHO command uses as arguments the results of the ACTIVE command (via textual substitution). When the macro is executed, WHO returns information on *each* process running in the system at the time the macro was executed. For example:

```
) ? ↲
PID:  1   @DPD0:CLI.PR
PID:  7   @DPD0:UTIL:MASM.PR
PID:  9   @DPD0:CLI.PR
PID:  27  @DPH0:LOG:POWERCHECK.PR
)
```

Convert Octal Numbers to ASCII Equivalents

)ASCII *octal-number-1 [...octal-number-n]* ⤶

Convert octal numbers to their corresponding ASCII values and display the results. The numbers you specify must be positive integers in the range 0 to 377.

ASCII *octal-number-1 [...octal-number-n]* ⤶

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) ASCII 46 ⤶
&
```

Converts octal 46 to its ASCII value, &, and displays its value.

ATTRIBUTES

Set or Display File Attributes

)ATTRIBUTES *filename [attributes-list]* ↲

Set or display a file's attributes. Enter *P, R,* and/or *W* for *attributes-list:*

A	File is *attribute-protected* (its attributes cannot be altered). Attribute A is set only by the system.
P	File is *permanent* (it cannot be renamed or deleted).
R	File is *read-protected* (it cannot be read).
W	File is *write-protected* (it cannot be modified).

If you enter ATTRIBUTES without a list of attributes, the CLI displays the current attributes for the file. If no attributes are set, the CLI displays this response:

```
<filename> NO ATTRIBUTE PROTECTION
```

NOTE: When you create a directory the system automatically assigns the permanence (P) attribute to the file. Other files do not have attribute protection before you assign it, which means that until then, you can delete, read, or modify those files. P must be turned off before a directory or any other permanent file can be deleted. When you set a file's attributes, the new attribute list replaces the current attributes.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/K	Clear attributes.

Examples

```
) ATTRIBUTES FORTRAN P↲
```

Makes the file named FORTRAN permanent.

```
) ATTRIBUTES FORTRAN ↲
FORTRAN P
```

Displays the attributes of the file FORTRAN.

BLOCK Stop a Process

) BLOCK *process-id* ↲

Stop a process from running. MP/AOS stops execution of the specified process until another process restarts it or the CLI UNBLOCK command is issued. If the process is already blocked, this command will have no effect. See Chapter 5 for more about blocking and unblocking processes.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) BLOCK 7 ↲
```

Blocks Process 7 from further processing by MP/AOS. This process will not be rescheduled until another process unblocks it by issuing the UNBLOCK command.

BOOT

Shut Down the Operating System

BOOT ⤶

Shut down the currently executing system. Before the system is shut down, BOOT closes all open I/O channels and dismounts all disk devices.

NOTE: *This command is valid only when issued from any CLI running in the initial process.*

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) BOOT ⤶
```

Shuts down the currently running system.

BYE Terminate the CLI

) BYE *[argument-list | message]* ↲

If you include a list of arguments or a message, the CLI returns it to the parent program (if any).

If you enter BYE when processing in the root CLI, Ozmos will shut down.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) BYE ↲
MP/AOS CLI TERMINATING
```

Terminates the CLI.

```
) BYE LOADER,EXTENDER ↲
```

Terminates the current version of the CLI and returns the argument list LOADER,EXTENDER to the parent program.

Chain to a New Program

) CHAIN *program-name [program-arguments]*⤶

Overwrite the CLI with the specified program and execute the new program in the same process. Any *program-arguments* you specify are passed to the new program. This command overwrites the currently executing CLI with the new program, and then executes it. The program swap level remains unchanged. The state of the currently executing CLI is not saved.

To pass input to the program, include the /I=*pathname* switch. This enables you to pass script files to the program. A script file contains instructions to achieve a specific end result without keyboard intervention.

Refer to Chapter 5 for more information about chain operations.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/I =*pathname*	Read program input from the specified file instead of from the console.
/O =*pathname*	Write the program output to the specified file instead of to the console. If you specify a simple filename that does not already exist, it will be created.

Examples

```
) CHAIN PROGRAM1 ↲
```

Loads PROGRAM1 into the CLI memory space and begins executing it.

```
) CHAIN ENVIRONMENT STARTUP ↲
```

Overwrites the CLI with the program ENVIRONMENT, to which the argument STARTUP is passed.

```
) CHAIN/I=FUEL/O=COST OIL_IMPORT.PR ↲
```

Loads OIL_IMPORT.PR into the CLI memory space and executes it. The file FUEL will be accessed for input in place of the keyboard, and the program output will be written to COST.

Set or Display Device Characteristics

) CHARACTERISTICS *[device]* ↓

Set or display the device characteristics for the character device you specify. Device characteristics control the way a device interprets input or sends output. You can set the characteristics for a character device in your system configuration by including the appropriate switches listed in the Command Specific Switches table. Notice that these switches may affect input only, output only, or both input and output.

Whenever you turn a characteristic on or off, the other characteristics for that device are unaffected. The characteristics you change remain in effect until you change them again, or until the system shuts down.

All characteristics except those set by the /CPL=*n* and /LPP=*n* switches are turned on or off. You turn on a switch or sequence of switches by preceding it with the /ON switch, and turn off a sequence by preceding it with /OFF. The /ON switch is optional as long as /OFF does not appear in the sequence of switches. /CPL=*n* and /LPP=*n* control line length and page length, respectively, and therefore can be set only to a different value (they cannot be turned on and off).

Any process may set the characteristics for any device. Once set, characteristics for any device remain in effect until changed again, or until the system shuts down.

If you enter CHARACTERISTICS without switches, the CLI displays the characteristics of the device you specify. If you do not specify a device, the CLI assumes you are referring to the user console. The default characteristics for a device depend on how the device was set up when the system was generated. See *MP/AOS System Generation and Related Utilities* (DGC No. 069-400206).

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/OFF	Turn off the characteristics named by the switches that follow this switch until the switch list ends or /ON switch occurs in the command line.
/ON	Turn on the switches that follow this switch until the switch list ends or /OFF appears. This is optional, because unless a switch you specify follows /OFF, the CLI turns it on.
	NOTE: *THE SWITCHES THAT FOLLOW ARE TURNED ON WHEN PRECEDED BY /ON AND OFF WHEN PRECEDED BY /OFF.*
/605X	When on, the Delete key erases the character preceding the cursor, and CTRL-U deletes the entire cursor line. This applies to consoles 6052 and 6053 only. /ECHO must be on.
/8BT	When on, leave all characters as 8-bit characters. When off, mask all input characters to 7 bits.
/BIN	Set Binary Mode on or off. When on, it permits an 8-bit character to be passed on input and disregards special characters on input and output.
/ECHO	When on, echo all characters read from the input to the output device; interpret special characters and echo control characters by prefixing them with an up-arrow. When Echo mode is off, control characters are still interpreted, but it is the programmer's responsibility to echo characters.
/EMM	When on, echo control characters exactly as input; control characters are not echoed as ↑x.
/ESC	Set Escape mode on or off. When on, the Escape character is interpreted as a CTRL-C CTRL-A sequence.
/ICC	When on, ignore control characters, except delimiters and characters interpreted by the system.
/LIST	Set List mode on or off. When on, CTRL-L (octal 14) is output as ↑L rather than as a Form Feed.
/NAS	Set non-ANSI standard on or off. When on: input Carriage Returns are converted to Line Feeds, while input Line Feeds are converted to Carriage Returns. Output Line Feeds are echoed as Carriage Return-Line Feed-Null.
/NED	When on, does not echo delimiters.
/ST	Set Software Tab Simulation mode on or off. When on, CTRL-I simulates a tab stop every 8th column on output. Binary mode (/BIN) must be off.
/UCO	When on, translate lowercase characters to uppercase on output.
	NOTE: *THE SWITCHES THAT FOLLOW CANNOT BE ENTERED WITH /ON OR /OFF.*
/CPL=*n*	Set the maximum number of characters per line to n for the specified device.
/LPP=*n*	Set the maximum number of lines per page to n for the specified device.
/I	Reset the user console to the characteristics it had when the current version of the CLI began executing. No argument or switches other than standard switches are allowed.
/P	Set the user console to the characteristics it had in the previous environment level. No argument or switches other than standard switches are allowed.
/RESET	Reset the characteristics of the specified device to the values they had at the time the system was booted. No switches other than standard switches are allowed.

Examples

```
) CHARACTERISTICS ↲
/LPP=24/CPL=80
/ON/605X/ST/UCO/ECHO
/OFF/BIN/ESC/LIST/NAS/8BT/NED/EMM/ICC
) CHARACTERISTICS/BIN ↲
) CHARACTERISTICS ↲
/LPP=24/CPL=80
/ON/605X/BIN/ECHO/ST/UCO
/OFF/ESC/LIST/NAS/8BT/NED/EMM/ICC
)
```

Displays the current characteristics for the user console, and then turns on the Binary mode. The other characteristics remain the same.

```
) CHARACTERISTICS/RESET @CON12 ↲
```

Resets console 12 to its initial characteristics.

```
) LEVEL ↲
LEVEL 1
) CHARACTERISTICS/P ↲
```

Resets the user console to the same characteristics as those in Level 0 (the previous environment level).

CLASS1 Set or Display CLASS1 Condition

) CLASS1 *[error-response]* ↓

Set or display the CLI response to CLASS1 exception conditions. CLASS1 exception conditions (described in Chapter 4) are caused by syntax errors and errors in commands that affect the CLI environment. These are usually more serious than CLASS2 exception conditions.

The CLASS1 command sets the CLI reaction to an error to IGNORE, WARNING, or ERROR, which is what you type for *[error-response].*

IGNORE	The CLI displays no message for a CLASS1 exception condition and continues processing, if possible.
WARNING	The CLI displays a warning message for a CLASS1 exception condition and continues processing, if possible.
ERROR	The CLI displays an error message for a CLASS1 exception condition and discards the input that is in the command buffer at the time the system encounters the mistake. The command buffer contains all input from the last prompt to the New-line character. This means that at the time the error is detected, the buffer might contain one or more commands and/or macros. If the error occurs while a macro is executing, the macro and any that invoked it will be terminated.

If you enter CLASS1 without an argument, the CLI displays the current error-response value (IGNORE, WARNING, or ERROR). ERROR is the initial value.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) CLASS1 WARNING ↲
) CLASS1 ↲
WARNING
```

Sets the CLASS1 value to WARNING, and verifies the change.

CLASS2 Set or Display CLASS2 Condition

) CLASS2 *[error-response]* ↵

Set or display the CLI response to CLASS2 exception conditions. CLASS2 exception conditions (described in Chapter 4) are caused by errors in commands that do not affect the current CLI environment. These are not as serious as CLASS1 errors, which could affect the CLI environment. The CLASS2 command sets the CLI reaction to IGNORE, WARNING, or ERROR, which you enter for *[error-response]*.

IGNORE	The CLI displays no message for a CLASS2 exception condition and continues processing, if possible.
WARNING	The CLI displays a warning message for a CLASS2 exception condition and continues processing, if possible.
ERROR	The CLI displays an error message for a CLASS2 exception condition 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 means that at the time the error is detected, the buffer might contain one or more commands and/or macros. If the error occurs while a macro is executing, the CLI will terminate the macro and any macros or commands that called it.

If you enter CLASS2 without an argument, the CLI displays the current error-response value (IGNORE, WARNING, or ERROR). WARNING is the initial value.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) CLASS2 ERROR ↲
) CLASS2 ↲
ERROR
```

Changes the original value of CLASS2 to ERROR, and verifies the change.

CONTROL Send IPC Message

) CONTROL *ipc-server-name message* ⮐

Send a control message to an interprocess communication (IPC) server. The CLI sends the message to the specified IPC server. An IPC server is a process that can receive IPCs. *MP/AOS System Programmer's Reference* (DGC No. 093-400051) describes interprocess communication.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) CONTROL SPREADER ON ⮐
```

Sends the message ON to the IPC server declared as SPREADER.

Copy Files

) COPY *destination-file source-file-1 [...source-file-n]* ↵

Copy one or more files to a destination file. If the destination file does not already exist, the COPY command creates a file and copies the source file(s) into it. The new file's specifications will depend on those of the first (or only) source file. When the source file is a disk file, the destination file will have the source file's specifications. The table below lists the default specifications for a new disk file that is copied from a peripheral device file.

File type	Text file
Element size	512 bytes
Time block	Time of last access and time of last modification are set to the current system time

Table 9.1 Peripheral device to disk file default specifications

Whenever you do a copy, the time and date of last access and of last modification for the destination file are set to the current system time.

NOTE: *The COPY command does not apply to magnetic tape devices.*

If the destination file already exists and is not a peripheral device, you must include either the /A or /D switch with the COPY command. /A appends the contents of the source file(s) to the destination file. /D deletes the destination file, creates a new file with the same name and specifications as the old destination file, and copies the source file(s) into the new file. If the destination is a device, the use of /A suppresses the printing of an initial Form Feed.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/A	Append the contents of the source file(s) to the contents of an existing destination file. When the destination file is a character device, /A suppresses the printing of an initial Form Feed.
/B	Set binary mode to on; the system will not interpret control characters when copying to or from a character device. This has no effect on disk-to-disk copies.
/D	Delete the destination file; create a new destination file with the same name and specifications as the old one; and copy the contents of the source file(s) into the destination file.

Examples

```
) COPY NUTRIENTS ORGANIC INORGANIC ↵
```

Copies ORGANIC and INORGANIC into a new file, NUTRIENTS, creating NUTRIENTS with the same specifications as ORGANIC.

```
) COPY/A PROGRAM MODULE1 MODULE2 ↵
```

Appends the contents of MODULE1 and MODULE2 to the contents of PROGRAM.

```
) COPY/D TEST QUESTION ↵
```

Deletes the contents of TEST and replaces them with the contents of QUESTION. TEST will be the same file type and have the same element size it had before the execution of this command.

Create a File

) CREATE *pathname* ↵

) CREATE/LINK *linkname link-contents* ↵

If you do not use switches with this command, the system creates an empty text file (TXT) with an element size of 1 block (512 bytes).

To create a link file (identified by the mnemonic LNK), include the /LINK switch, replace *pathname* with the name of the link, and enter another argument, the contents of the link. The link can contain a pathname or partial pathname, or any other text stream.

Then whenever you include the *linkname* as part of a pathname, the CLI resolves the *linkname* to the link contents (if the contents are resolvable). The maximum length for *link-contents* is 63 bytes.

NOTE: *Directories are created by the system with the permanence (P) attribute set. This means that directories cannot be renamed or deleted until you turn off this attribute. For more information, refer to the RENAME and DELETE commands.*

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/DELETE	Delete any existing file that has the specified name.
/DIRECTORY	Create a directory.
/ELEMENTSIZE=*n*	Set the file element size to *n* disk blocks of 512 bytes.
/I	Take the contents of the file from subsequent keyboard input. The last line must contain a single) or be terminated by a CTRL-D to indicate that there is no more input.
/LINK	Create a link. This switch cannot be used in combination with any switches except /DELETE.
/M	Take the input to the file from subsequent lines in the current macro body. The last line must contain a single).
/TYPE=*n*	Create a file of the type indicated, where *n* is one of the file types listed in Table 9.2.

Identifying Mnemonic	File Type
BPG	MP/OS format bootable program*
BRK	Program break file
DIR	Directory
IDF	MP/ISAM file
IXF	MP/ISAM index file
LIB	Library
LNK	Link file
LOG	System log file**
MBS	MP/BASIC save file
OBF	Object file
OLF	Overlay file
PRG	Program file
PST	Permanent symbol table (used by assembler)
STF	Symbol table file
TXT	Text file
UDF	User data file

Table 9.2 MP/AOS file types

**Currently not bootable under MP/AOS.*

***LOG can be shown via FILESTATUS but not created.*

Examples

```
) CREATE/DIR PAINTING ⤶
) CREATE MATISSE ⤶
```

Creates a directory called PAINTING and a file called MATISSE in the current working directory.

```
) CREATE/I SWITCH.CLI ⤶
)) RENAME %1% /TEMP ⤶
)) RENAME %2% %1% ⤶
)) RENAME TEMP %2% ⤶
))) ⤶
```

Creates a macro file called SWITCH.CLI and accepts input from the console.

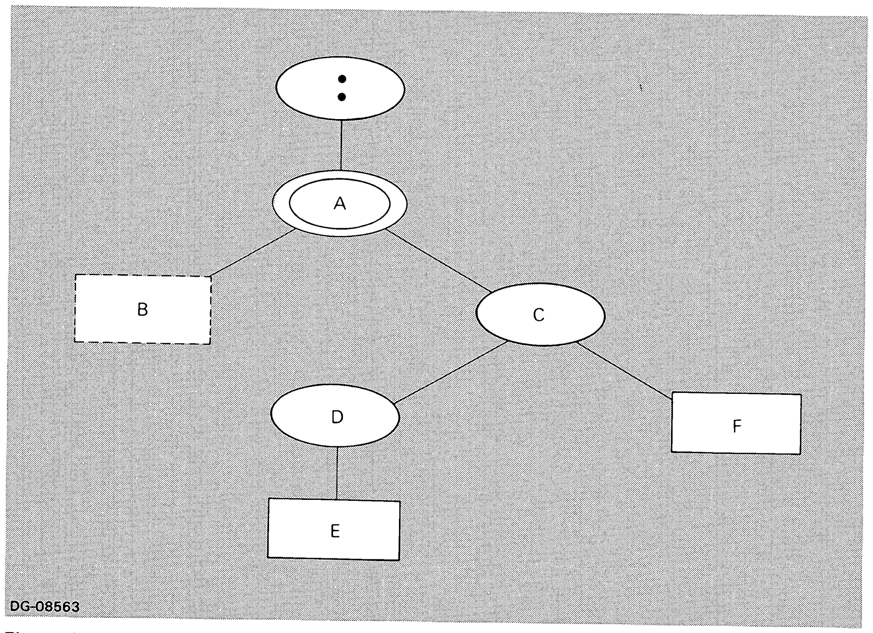


Figure 9.1 Link examples

```
)DIRECTORY ⤶
:A
) CREATE/LINK B C:D ⤶
) DIRECTORY B:G ⤶
) DIRECTORY ⤶
:A:C:D:G
```

Creates a link file named B containing the partial pathname C:D. The CLI resolves B:G to C:D:G, thereby changing the working directory to :A:C:D:G.

```
) CREATE/TYPE=99/ELEMENTSIZE=2 SYNTHESIS ⤶
) FILESTATUS/ASSORTMENT SYNTHESIS ⤶
DIRECTORY :PROTEIN
SYNTHESIS   99   7-FEB-81  11:15:29  0
```

Creates an empty, user-defined file (identified as file type 99) named SYNTHESIS, for which space will be allocated in multiples of 2 disk blocks (1024 bytes), and displays assorted information about the new file.

CURRENT Display Current Values for CLI Environment

```
) CURRENT↲
```

This command accepts no arguments and displays the environment in which you are currently processing your data. The following table lists the parameters that define the CLI environment and the corresponding CLI commands that set and display these values.

Corresponding CLI Command	Parameter
CHARACTERISTICS	Console characteristics
CLASS1	CLASS1 error response
CLASS2	CLASS2 error response
DIRECTORY	Working directory
LEVEL	Current environment level
MEASURE	Elapsed time reporting mode setting
PROMPT	Prompt setting
SEARCHLIST	Searchlist
STRING	String values
TRACE	Trace mode setting

Table 9.3 The CLI environment

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) CURRENT ↲
```

```
LEVEL:              0
DIRECTORY:          :ORGANIC:ESTER
SEARCHLIST:         @DPDO:BIOCHEM,@DPDO:INORGANIC
CHARACTERISTICS:    /CPL=80/LPP=24
                    /ON/605X/ECHO/ST
                    /OFF/BIN/ESC/LIST/NAS/UCO/8BT/NED/EMM/ICC
CLASS1:             ERROR
CLASS2:             WARNING
MEASURE MODE:       OFF
TRACE MODE:         OFF
STRING BUFFERS:
PROMPT:
```

Displays the current CLI environment. Refer to Chapter 4 for a description of the CLI environment and how to modify it.

DATE Set or Display System Date

DATE *[dd-mon-yy | mm-dd-yy | mm/dd/yy | mm dd yy]*

Entering the DATE command without arguments displays the current system date. To set the date, enter the date in one of the above formats with these substitutions:

dd	one- or two-digit day
yy	two-digit year
mon	first three letters of month name
mm	two-digit month

No matter which format you use to enter the date, the system displays it as *dd-mon-yy.*

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) DATE↵
22-MAR-81
```

Displays the current system date.

```
) DATE 3/25/81 ↵
) DATE ↵
25-MAR-81
```

Sets and then displays the system date.

Execute Program through the Symbolic Debugger

) DEBUG *program-name [program-arguments]* ↵

The program you specify must be an executable program file. It is blocked before it begins execution, allowing the Symbolic Debugger to examine or modify its state. See the Symbolic Debugger in *MP/AOS Process Utilities,* (DGC No. 069-400205).

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/F=*filename*	Take input from the specified file for debugger macro definitions. When the file's contents have been exhausted, take debugger input from the keyboard.

Examples

```
) DEBUG HI_TECH.PR ↵
```

Executes the HI_TECH.PR program file and invokes the Symbolic Debugger to debug the program.

```
) DEBUG MYPROG TEXTFILE ↵
```

Executes MYPROG and invokes the Symbolic Debugger. TEXTFILE is passed to MYPROG as an argument.

DECIMAL Convert Octal Number to Decimal

) DECIMAL *octal-number*⤶

Translate an octal number to its corresponding decimal value. The octal number you specify must be an unsigned integer in the range 0 to 37,777,777,777.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) DECIMAL 100 ⤶
64
```

Returns the decimal equivalent of the octal number 100.

Delete One or More Files

) DELETE *pathname-1 [...pathname-n]* ⤶

Unless you include the /DIR switch, DELETE will remove a directory file only if it is empty. The /DIR switch will delete the directory and any subtrees it contains.

To delete a permanent file, turn off the permanence (P) attribute using the ATTRIBUTES command, then delete the file. The /DIR switch does this automatically for directory files.

NOTE: *This command accepts templates in place of pathnames. If you enter a template or partial pathname, DELETE uses only the working directory, not the searchlist, to find the file(s).*

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/C	Display the name of each file to be deleted and allow the user either to confirm the deletion by typing Y, or to prevent the deletion by typing N or New-line.
/DIR	Delete the specified directory and its sub-tree, if any.
/V	Verify each deletion with a list of files deleted.

Examples

```
) DELETE/C FLOWERS ↵
FLOWERS?↵
FILE NOT DELETED
```

Since a New-line was typed in response to the confirmation request, FLOWERS was not deleted.

```
) DELETE/V GRAPHICS WATERCOLORS ↵
DELETED GRAPHICS
DELETED WATERCOLORS
```

Confirms that two files, GRAPHICS and WATERCOLORS, have been deleted.

```
) DELETE/DIR ARTWORK ↵
```

Deletes directory ARTWORK and its subtree.

Set or Display Working Directory

) DIRECTORY *[pathname]* ⤶

You can set the working directory to any directory in the file system. If you enter DIRECTORY without an argument and switches, the CLI displays the current pathname of the working directory.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/I	Set the working directory to the session's initial working directory when no argument appears in the command. When an argument is included, set the working directory to that subdirectory of the initial working directory. In either case, the command echos the new directory.
/P	Set the working directory to the working directory in the previous environment level and echo the new directory name. (No argument is allowed.)

Examples

```
) DIRECTORY TOPICS ⤶
) DIRECTORY ⤶
@DPDO:MANUALS:TOPICS
```

Sets the working directory to a subordinate directory named TOPICS and displays its pathname.

```
) DIRECTORY/I LSI ⤶
@DPDO:LSI
```

Changes the working directory to the directory LSI, which is subordinate to the initial working directory, and echoes its pathname.

```
) LEVEL ↵
LEVEL 1
) DIRECTORY/P ↵
@DPD0:NETWORKS
```

Displays the environment level and then changes the current working directory to the working directory of the previous environment level, Level 0.

Display Disk Information

DISKSTATUS

) DISKSTATUS *[diskname]* ↲

Display information about the space available on the specified disk and provide the disk's error history.

If you enter this command without an argument, the CLI displays information about the disk containing the current working directory. DISKSTATUS without switches displays the number of disk blocks available and number of blocks used.

If the disk has experienced errors, the CLI lists the appropriate error message (one of the following):

```
BAD PRIMARY LABEL BLOCK
BAD SECONDARY LABEL BLOCK
BAD PRIMARY MDV BLOCK
BAD SECONDARY MDV BLOCK
# RECOVERABLE ERRORS
# UNRECOVERABLE ERRORS
```

If the CLI displays one of the first four error messages, refer to the FIXUP utility in *MP/AOS System Generation and Related Utilities* (DGC No. 069-400206).

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/B	Display the space available and space in use in bytes instead of blocks.
/F	Indicate how many files can be created on the disk. This number is set when the disk is initialized.

Examples

```
) DISKSTATUS ↵

@DPH0

22503   BLOCKS AVAILABLE
2073    BLOCKS IN USE
```

Displays the amount of space that is available and the amount that is currently used on @DPH0, the disk containing the working directory.

```
) DISKSTATUS/F ↵

@DPH0

22503   BLOCKS AVAILABLE
2073    BLOCKS IN USE
1016    FILES CAN BE CREATED
```

Displays the same information about the @DPH0 disk as in the last example, but also indicates the maximum number of files that can be stored on the disk.

Prepare a Disk for Removal

) DISMOUNT *directory-device* ↵

The disk or diskette that you specify as *directory-device* is disabled from further file system activity, and is prepared for removal from the system without loss of data. This command will fail if one or more directories on the disk are in the searchlist, or if the working directory is on this disk.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) DISMOUNT @DPD4 ↵
```

Prepares disk @DPD4 for removal from the system.

!ELSE Execute Alternative Commands (Pseudomacro)

[!ELSE]

Executes an alternative sequence of input lines for an !EQUAL or !NEQUAL pseudomacro block whose condition is not true. Pseduomacros are explained in Chapter 6.

Follow !ELSE with the command lines to be executed conditionally. If the condition stated by the corresponding !EQUAL or !NEQUAL is true, the CLI ignores the !ELSE sequence and continues processing after the block's !END line. However, when the condition is not true, the CLI ignores the input lines between the conditional statement and !ELSE; processing continues immediately following the !ELSE line.

Example

(This example shows CLI commands written in a macro.)

```
WRITE FUEL COST FOR THE CURRENT BUDGET YEAR
[!EQUAL OIL [!READ PLOT GRAPH FOR OIL OR GAS?]]
XEQ OIL.RPT.PR
[!ELSE]
XEQ GAS.RPT.PR
[!END]
```

The macro shown here executes one of two possible programs, depending on how you respond to the !READ statement. If you enter OIL, the CLI executes a program that graphs oil usage. If you enter anything else, the conditions stated by !EQUAL will not be true, so the CLI will execute the lines following !ELSE.

End a Conditional Block (Pseudomacro)

[!END]

Marks the last line of an !EQUAL or !NEQUAL block. Include this pseudomacro with every pseudomacro block you use. If you have nested !EQUAL or !NEQUAL blocks, an !END completes the innermost open block. Pseudomacros are explained in Chapter 6.

Example

(This example shows lines from a macro.)

```
[!EQUAL JAN %1%]
 XEQ QTR1.PR
 [!EQUAL APR %2%]
 XEQ QTR2.PR
 [!END]
[!ELSE]
 XEQ QTR3.PR
 XEQ QTR4.PR
[!END]
```

The sequence here contains two !EQUAL blocks, one nested inside the other. The second !EQUAL is nested in the first. Notice that the first !END pseudomacro completes the innermost !EQUAL block. The !ELSE line and following lines are executed only if the first !EQUAL condition is not true.

!EQUAL Test for String Equivalence (Pseudomacro)

[!EQUAL *argument-1,argument-2*]

Compares two arguments and continues processing according to whether or not the !EQUAL condition is true.

An !EQUAL block consists of the !EQUAL pseudomacro, the input lines the CLI should execute when the two arguments match, and the !END pseudomacro to complete the block. You may want to include the !ELSE pseudomacro in your sequence of input lines, too. !ELSE executes an alternative sequence of lines when the !EQUAL condition is not true. See the command description for !ELSE.

You must enter two arguments for !EQUAL to compare. !EQUAL compares both arguments character by character. If either argument is null, use commas as separators to indicate this fact.

If the arguments are identical, the CLI processes the input lines that you have included between the !EQUAL statement and the corresponding !END (or the !ELSE, if you include it). If the two arguments are not identical, the CLI skips to the block's !END statement (or !ELSE block, if included), and then continues processing.

Examples

```
[!EQUAL [!DATE] 1-JAN-81]
 XEQ PROG1 LOGISTICS
[!END]
```

If the current system date is 1-JAN-81, this macro executes PROG1. If the two dates are not identical, the CLI skips execution of PROG1 and continues processing after the !END line.

```
XEQ/S BARNAC %1%
[!EQUAL,[!STRING],]
 XEQ REPORT.PR
 [!ELSE]
 DEBUG BARNAC %1%
[!END]
```

The !EQUAL statement compares String Buffer 1, which contains the termination string from the BARNAC program, to the null string. If they are equal, the CLI executes REPORT.PR; otherwise, the CLI invokes the Symbolic Debugger to debug BARNAC. Note that the commas in the !EQUAL statement are crucial—the use of spaces as delimiters here would cause the wrong effect if the value of !STRING is null.

ERCODE

Display Most Recent CLI Error Code

) ERCODE ⤶

Display the five-digit error code resulting from the last CLI command. If the previous CLI command was successfully executed, this command returns a zero. If an error occurred, ERCODE returns one of the five-digit error codes listed in Appendix A. ERCODE reflects the error that occurred on the last command even if exception handling was set to IGNORE and no error message was displayed.

The octal value returned by ERCODE can be used either as a string or as an argument to an arithmetic operation.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) DATE ⤶
12-APR-81
) ERCODE ⤶
0
```

Displays the system date without generating an error, so ERCODE returns a zero.

The following lines excerpted from a macro illustrate the use of !ERCODE.

```
    [!EQUAL 0 [!ERCODE]]
    COPY/A LEXICON %1%
    [!ELSE]
    XEQ DECODE %1%
    [!END]
```

When ERCODE equals 0, the macro appends the first macro call argument to file LEXICON. Otherwise, the macro executes the DECODE program.

```
) XEQ ERROR.PR [!UDIVIDE [!ERCODE] 1024] ↲
```

Executes ERROR.PR and passes to it the result of the error code divided by 1024.

EXECUTE

Swap in a New Program

) EXECUTE *program-name [program-arguments]*⤶

Write the CLI out to disk and read a program into memory to run in its place. Any arguments you enter are passed to the specified program. MP/AOS swaps out the CLI and saves its processing status on disk, the program swap level is incremented by one, and the specified program executes in the same process address space as the CLI, but at the incremented program swap level. (You can nest program levels to Level 8.)

When processing of the program ends, control is returned to the parent program, the CLI, which executes at its original program swap level. The CLI will report any exception conditions returned by the executed program or by the MP/AOS system. Refer to Chapter 5 for further information about program execution.

NOTE: *EXECUTE functions identically to XEQ.*

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/I=*pathname*	Read program input from the specified file instead of from the console.
/M	Read program input from the currently executing macro instead of from the console. The last input line must be a single).
/O=*pathname*	Write the program output to the specified file instead of to the console. If you give a filename that does not already exist, it will be created.
/S	Write the program termination message to String Buffer 1 instead of to the console. This switch cannot be used with /L(=).

Examples

```
) EXECUTE PROGRAMX ↲
```

Swaps out the CLI and then executes PROGRAMX in its place. MP/AOS saves the processing status of the CLI on disk.

```
) EXECUTE MASM PROGY ↲
```

Invokes the Macroassembler, passing it the name PROGY, the source file to be assembled.

```
) EXECUTE/S CONSTRUCT.PR ↲
```

Executes program CONSTRUCT.PR. The program termination message will be written to String Buffer 1 for later use.

```
) EXECUTE/I=RESOURCES/O=TIMETABLE PROJECT_PLAN.PR ↲
```

Invokes the program named PROJECT_PLAN.PR, using the RESOURCES file for input. The program will write its output to TIMETABLE, rather than to the console.

EXPLODE

Space Out a String

) EXPLODE *string* ↓

Convert all spaces, tabs, and commas in a character string to commas, and then place a space between every two characters (including commas) in the new string.

Exploding a string is useful for the purpose of removing certain characters from the string while retaining others. You may want to use the exploded characters as arguments to other commands or macros. See Chapter 6 for an explanation of exploding character strings.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) EXPLODE NEGZS# 0,1,SEZ ↓
N,E,G,Z,S,#,,,0,,,1,,,S,E,Z
```

Explodes the included character string. The space and commas are converted to commas before the characters are separated.

FILESTATUS

List File Status Information

) FILESTATUS *[pathname-1 [...pathname-n]]* ⤶

Besides listing the filename and pathname of the file's parent directory, this command displays information according to the switches you include. You can enter a filename template in place of a pathname.

NOTE: *The searchlist is not referenced by this command.*

Be aware that FILESTATUS does not resolve a linkname to the link it contains. If you request information about a link file with the /ASSORTMENT switch, the CLI displays the linkname and its contents.

If there is not enough space in memory to do a sort using either /SORT or /SORT=*field*, the CLI will not attempt the sort; instead, it will list the files in unsorted order.

If you omit arguments when entering the command, the CLI displays the information for all the files in the working directory. If you do not include switches, the parent directory pathname and the filenames are displayed.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/AFTER	Must be used with the /TLA or /TLM switch (described below) to display information that applies after a specified time.
/ASSORTMENT	Display the file type, the time and date the file was last modified, and the file length in bytes.
/BEFORE	Must be used with the /TLA or /TLM switches to display information that applies before a specified time.
/DLA	Display the date the file was last accessed.
/DLM	Display the date the file was last modified.
/ELEMENTSIZE	Display the number of disk blocks in a file element for this file.
/LENGTH	Display the file length in bytes.
/NHEADER	Do not display the pathname header(s) for the parent directories. List each file by pathname, not filename.
/SORT	Display filenames and associated information in alphabetical order, according to filename.
/SORT=*field*	Order the information to be displayed by one of the fields listed in Table 9.4, Sorting by Fields.
/TLA	Display the date and time the specified files were last accessed.
/TLA = *date[time]* | *time-period*	Display filenames for only those files last accessed: (include /AFTER) on or after a specific date and time or during a given period of time; or (include /BEFORE) before a specific date and time or period of time.*
/TLM	Display the date and time the specified files were last modified.
/TLM = *date[time]* | *time-period*	Display information for only those files last modified: (include /AFTER) on or after a specific date and time or during a given period of time; or (include /BEFORE) before a specific date and time or period of time.*
/TYPE	Display the type of each requested file.
/TYPE=*type*	Selects all files of the specified type. Replaces type with either a three- character mnemonic or a number. See Table 9.5 MP/AOS file types.

* Enter *[date][time]* in the format *[dt-mon-yy][:hh:mm:ss]*; or time-period in the format *-da[:hh[:mm]]*, where *dt* is the date, *da* is the number of days prior to today, *mon* is the month, *hh* is hours, *mm* is minutes, and *ss* is seconds.

In the *[date][time]* format you can specify the date only, the time only, or both together. If you omit the date, today's date is assumed; if you omit the time, midnight is assumed.

For example, enter *12-JAN-81:10:15:00* for 10:15 a.m., January 12, 1981; or *-1:1* for the last 25 hours.

You can display information about files in alphabetical order by filename using the /SORT switch. However, you can display the same information sorted in a different way by using the keyword form of this switch, /SORT=*field*. The following table lists the fields on which you can sort.

/SORT=	Sort File List by:
ELEMENTSIZE	Element size - shortest to longest
LENGTH	File length - shortest to longest
TLA	Time & date last accessed - earliest to latest
TLM	Time & date last modified - earliest to latest
TYPE	File type

Table 9.4 Sorting file list by fields

Identifying Mnemonic	File Type
BPG	Bootable program*
BRK	Program break file
DIR	Directory
IDF	MP/ISAM data file
IXF	MP/ISAM index file
LIB	Library
LNK	Link file
LOG	System log file**
MBS	MP/BASIC save file
OBF	Object file
OLF	Overlay file
PRG	Program file
PST	Permanent symbol table (used by assembler)
STF	Symbol table file
TXT	Text file
UDF	User data file

Table 9.5 MP/AOS file types and mnemonics accessible via the CLI

**Currently not bootable under MP/AOS.*

***LOG may be shown via FILESTATUS but not created.*

Examples

```
) FILESTATUS/SORT ↵

DIRECTORY @DPD0:ED

COOP    DIRECT15  EARLY    LATE
```

If you use the simple form of the SORT switch, the files are sorted by filename in alphabetic order.

```
) FILESTATUS/SORT=TLA ↵

DIRECTORY @DPD0:ED

EARLY    LATE   DIRECT15    COOP
```

Displays the filenames contained in the working directory in the order of time of last access. EARLY was accessed least recently and COOP most recently.

```
) FILESTATUS/ASSORTMENT
DIRECTORY @DPDO:ED

DIRECT15      UDF     5-DEC-80      15:14:30     3162
COOP          UDF    30-DEC-80      14:16:30      976
EARLY         TXT     3-NOV-80      13:11:26     1340

LATE          DIR     1-DEC-80      11:03:12     4608
```

This command line displays filename, file type, date and time when the file was last modified, and file length in bytes for each file in the current working directory, @DPDO:ED.

```
) FILESTATUS/TYPE=DIR/NHEADER ⤶

@DPDO:ED:LATE
```

Displays the names of all files in the working directory that are directory files (DIR). Notice that the file is listed by pathname instead of filename.

```
) DATE ⤶
2-JAN-81
) FILESTATUS/AFTER/TLM=—10/TLM ⤶

DIRECTORY @DPDO:ED

COOP    30-DEC-80    14:16:30
```

Displays today's date, and then requests a listing of all files modified in the last 10 days, and the time and date they were modified.

HELP Explain a CLI Command or Topic

) HELP *[command | *topic]* ↓

This command provides information about the command or topic you specify.

HELP can always provide some information on a CLI command. To request a detailed description of a command, include the /V switch. To review a list of CLI topics for which you can request HELP information, type HELP without an argument.

If the CLI does not recognize the line you enter, it displays the message:

```
item   -UNKNOWN
```

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/V	Explain the specified command in detail.

Examples

```
) HELP XEQ ↓
XEQ    - REQUIRES ARGUMENT(S)
  SWITCHES: /1= /2= /L(=) /TIMEMODE /I= /M /O= /S

FOR MORE HELP TYPE
HELP/V XEQ
```

Displays information about the XEQ command. Notice that entering HELP with the /V switch will display further information about the XEQ command.

```
) HELP *macro ↓
```

Requests HELP information about macros.

INFORMATION

Display Executable File Information

) INFORMATION *program-pathname* ↓

This command describes the kind of file, such as a program file.

Include the switches to display such information as the starting address, length of the impure or pure memory, and the revision number.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/I	Display the impure memory starting address and length.
/P	Display the pure memory starting address and length.
/R	Display the revision number.

Example

```
) INFORMATION CLI.PR ↓
CLI.PR    PROGRAM FILE
```

Indicates that CLI.PR is a program file.

LEVEL Display CLI Environment Level Number

) LEVEL ↲

The CLI environment is initially at Level 0. The PUSH and POP commands control the CLI environment level. The PUSH command increments the level number by one, and the POP command decrements the level number by one. The maximum PUSH level depends on the current use of system resources. Level 0, however, is the last level to which you can POP.

Refer to the command descriptions for PUSH and POP. Do not confuse this command with SWAPLEVEL, which examines the program swap level.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) LEVEL ↲
LEVEL 0
```

Indicates that the current CLI environment is at Level 0.

```
) PUSH ↲
) LEVEL ↲
LEVEL 1
) POP ↲
) LEVEL ↲
LEVEL 0
```

Pushes from Level 0 to Level 1, and then pops to Level 0, again. Read Chapter 4 for a discussion of CLI environments.

MEASURE

Report Elapsed Time

) MEASURE *[ON | OFF]* ↵

Display the time used to execute each CLI command you enter.

This command turns the Elapsed Time Reporting mode on or off. By default MEASURE is off. When you turn MEASURE on, the CLI includes with each command response the time (in seconds) used to execute the command. Entering MEASURE without an argument displays its current setting (ON or OFF).

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) MEASURE ↵
OFF
```

Reports that the Elapsed Time Reporting mode is currently off.

```
) MEASURE ON ↵
) DATE↵
31-APR-81
01
```

Turns on the elapsed time reporting mode, and then displays the system date with the time used to execute the DATE command, 1 second. (Elapsed time is never shown in units smaller than seconds.)

MESSAGE Display Error Code Text

) MESSAGE *errorcode-1 [...errorcode-n]* ⤶

Display the text message that corresponds to the specified error code. There is a text message for each error code in the system.

NOTE: *MESSAGE interprets the codes you enter as octal integers, unless you include the /D switch, which indicates to the CLI that the arguments are decimal codes.*

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/D	Interpret the arguments as decimal integers instead of octal integers.

Example

```
) MESSAGE 41010 ⤶
41010 UNKNOWN COMMAND, MACRO, OR PROGRAM NAME
```

Displays the text message for error code 41010.

MOUNT

Make a Directory Device Available to System

) MOUNT *directory-device [disk-id]* ↵

You should replace *directory-device* with the unit name of the disk or diskette you want mounted. Table 9.6, below, lists the disks that are available to the MP/AOS system.

The second argument, *disk-id*, is optional but can be used to be sure that you have the right device. If you include it, the system will check the specified device to see if it has a matching label. If they match, the device is mounted; if they do not match, the device is not mounted. Whenever a device is successfully mounted, the CLI confirms the fact by displaying its label.

Refer to *MP/AOS System Generation and Related Utilities Programmer's Reference* (DGC No. 069-400206) for more information about preparing directory devices to be used by the system.

Unit Name	ECLIPSE	microNOVA
DPG	20 Mbyte cartridge disk	
DPD	10 Mbyte cartridge disk 315 Kbyte diskette	10 Mbyte cartridge disk
DPH	12.5/25 Mbyte fixed disk 1.25 Mbyte diskette	12.5/25 Mbyte fixed disk 1.25 Mbyte diskette
DPF	50/95/190 Mbyte cartridge disk	
DPX		315 Kbyte diskette

Table 9.6 Disk and diskette unit names

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) MOUNT @DPD3 ↵
MY_DISK
```

Makes the @DPD3 disk directory structure available to the operating system and confirms that it has been mounted by displaying the disk's label, MY_DISK.

```
) MOUNT @DPD4 DISK_1 ↵
DISK_1
```

Checks the @DPD4 disk device for a label named DISK_1, and then it confirms that it is in fact the requested device. This device is now available to the system.

NAME

Remove Final Extension

) NAME *argument-1 [,...,argument-n]* ↵

List the arguments without the final extension.

NOTE: *If an argument contains more than one period, only the last extension is removed.*

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) NAME LOOMA.FR FOO.SR.BU ↵
LOOMA FOO.SR
```

Displays the two arguments without their final extensions. Notice that FOO.SR.BU is displayed as FOO.SR because only the final extension is dropped.

!NEQUAL Test for String Non-Equivalence (Pseudomacro)

[!NEQUAL *argument-1 argument-2*]

Compares two arguments and continues processing according to whether or not the !NEQUAL condition is true.

The !NEQUAL statement must be completed with the !END pseudomacro. You can include commands between !NEQUAL and !END to be conditionally executed. !NEQUAL, the included commands, and !END make up an !NEQUAL block.

If the arguments are not identical, the CLI executes the commands in the block. If the arguments match, the condition is not true, so the CLI skips the rest of the block and begins processing after !END. Note that it is important to use commas as delimiters between arguments to !NEQUAL, especially in those cases where an argument may evaluate to a null string.

You may want to include an !ELSE block within the !NEQUAL block to execute an alternative sequence of lines when !NEQUAL is not true. If !NEQUAL is true, the CLI ignores the !ELSE block. When !NEQUAL is not true, the CLI skips to !ELSE and processes the block of lines following it.

Examples

```
XEQ/S REVENUE.PR %1%
[!NEQUAL 99,[!STRING]]
XEQ PROFIT.PR
[!END]
```

Executes program REVENUE.PR and writes its termination message to String Buffer 1. The !NEQUAL statement compares the contents of String Buffer 1 to 99. As long as they are not equal, the CLI continues processing by executing program PROFIT.PR. Otherwise, the CLI skips to after !END.

```
[!NEQUAL FY85,%1%]
XEQ BUDGET.PR %1%
[!ELSE]
WRITE SORRY, NO BUDGET PLANNED YET
[!END]
```

Compares the text string FY85 to the first argument in the macro call. As long as they are not equivalent, the CLI passes the first argument to program BUDGET.PR and executes the program. If they are equal, the CLI skips to after !ELSE and displays the message SORRY, NO BUDGET PLANNED YET.

OCTAL

Convert Decimal Number to Octal

) OCTAL *decimal-number* ⤶

Convert an unsigned decimal number to its octal equivalent and display the result. You can convert any decimal integer in the range 0 to 4,294,967,295.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) OCTAL 10 ⤶
12
```

Converts 10 in decimal to octal and displays the result.

PATHNAME

Display a Complete Pathname

) PATHNAME *partial-pathname* ↵

This command displays a complete pathname for the partial pathname you specify. Both pathnames and partial pathnames are discussed in Chapter 3.

If the file cannot be found in the working directory or searchlist directories, the CLI responds with an error message.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) DIRECTORY ↵
@DPDO:NETWORK
) PATHNAME MULTIPLEXORS ↵
@DPDO:TELECOM:MULTIPLEXORS
```

Displays the current working directory, and then displays the complete pathname for the MULTIPLEXORS file. This file is not in the current working directory. The system, therefore, looked through the directories in the searchlist to find the file.

```
) PATHNAME FREQUENCIES ↵
WARNING: FILE DOES NOT EXIST, FILE FREQUENCIES
```

Displays an error message for this command line. The system cannot find file FREQUENCIES in either the working directory or the searchlist directories.

```
) PATHNAME =DEVICE:FEEDBACK ↵
@DPDO:NETWORK:DEVICE:FEEDBACK
```

Displays the complete pathname for the file identified by the partial pathname =DEVICE:FEEDBACK. The system limits its search to the working directory, as designated by the = prefix. See Chapter 3 for more about pathname prefixes.

PAUSE

Delay CLI

) PAUSE *time-period* ⤶

The CLI pauses for the time you specify. *Time-period* is expressed in seconds or tenths of seconds in the range 0 to 3456000.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) PAUSE 30 ⤶
```

Pauses all CLI activity for 30 seconds.

```
) PAUSE 30.5 ⤶
```

Pauses the CLI for 30.5 seconds.

POP Return to Previous CLI Environment

) POP ⤶

This command decrements the current environment level number by one and returns you to that level, restoring its environment values. The environment you are leaving is not saved.

NOTE: *Level 0 is the last level to which you can POP. When you log on to the CLI, Level 0 is the initial environment level.*

Table 9.7 lists the parameters that define the each environment and the commands that set and display those values.

Corresponding CLI Command	Parameter
CHARACTERISTICS	Console characteristics
CLASS1	CLASS1 error response
CLASS2	CLASS2 error response
DIRECTORY	Working directory
LEVEL	Current environment level
MEASURE	Elapsed time reporting mode setting
PROMPT	Prompt setting
SEARCHLIST	Searchlist
STRING	String values
TRACE	Trace mode setting

Table 9.7 The CLI environment

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/V	Verify the new environment level by displaying the level number.

Examples

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

Displays the current level, Level 1, and then POPs, verifying the new level as Level 0.

PREVIOUS Display Previous CLI Environment Values

) PREVIOUS ↲

This command only shows the previous CLI environment level settings. It does not change the current environment level, nor does it change the current environment settings. If you enter this command while in Level 0, it will return an error. Refer to Chapter 4 for a discussion about CLI environment values and levels.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) LEVEL ↲
LEVEL 1
) PREVIOUS ↲
LEVEL:                 0
DIRECTORY:              :THERMO
SEARCHLIST:             :MATERIALS, :STRUCTURES
CHARACTERISTICS:       /CPL=80/LPP=24
                       /ON/ST/ECHO/605X
                       /OFF/NAS/ESC/LIST/BIN/UCO/8BT/NED/EMM/ICC
CLASS1:                ERROR
CLASS2:                WARNING
TRACE MODE:            OFF
MEASURE MODE:          OFF
STRING BUFFERS:
                       EQUILIBRIUM



PROMPT:                TIME
```

Displays the current environment level as Level 1, and then displays the environment settings for Level 0.

PRIORITY

Set or Display the Priority of a Process

)PRIORITY *[process-id] [priority]* ↓

Returns or sets the process priority of the indicated process. If no process identity number (PID) is specified, the command returns the priority of the CLI the command is issued from.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) PRIORITY ↓
3
)
```

The priority of the currently executing CLI is 3. (Third from the highest.)

```
) PRIORITY 27 127 ↓
```

Set the priority of process 27 to a very low priority.

PROCESS Create a Concurrent Process

) PROCESS *program-name [argument-1 [...argument-n]]* ⤶

Create another process to run in parallel with the CLI. The process you create runs concurrently with the CLI, executing the program you specify at Program Swap Level 1. Arguments 1 through *n,* if included, are passed to the program. The new process is independent from the CLI and the other processes in the system. It executes within its own address space and bears no hierarchical relationship to the process that creates it.

MP/AOS assigns each process an identification number (PID), which you can use for further process management commands.

The environment for a new process is defined by these parameters:

- searchlist
- working directory
- I/O channels
- maximum number of channels the process can open
- process priority
- maximum number of 1K-word pages the process can use
- maximum number of attached segments (excluding pure and impure segments automatically allocated by the system)
- maximum number of Task Control Blocks (TCBs) available to the process
- maximum number of overlay nodes for the process

All but the searchlist and the working directory can be set with the PROCESS command switches. The searchlist and working directory for the new process will be the same as the current values for the CLI. If you do not include switches, the corresponding parameters will have the same value as in the CLI. (Default process environment values for the initial CLI are listed in Table 9.8.) Two environment values are not passed to the new process. If you do not specify the /I and /O switches, Channels 0 and 1 will be closed. Chapter 5 discusses processes and process management.

Parameter	Value
Priority	0
Memory	
Maximum number of 1 K-word pages	32 pages
Maximum number of attached segments	0 (segments)
Maximum number of overlay nodes	2 (nodes)
Tasks	
Maximum number of TCBs	3
Channels	
Input channel	@TTI
Output channel	@TTO
Maximum number of channels	15
Searchlist	:
Working Directory	:

Table 9.8 Environment values for the initial MP/AOS process

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/CHANNELS=*n*	Limit the process to opening a maximum of *n* channels.
/DEBUG	Enter the Symbolic Debugger to debug the specified program.
/I=*pathname*	Open the file identified by *pathname* for input from Channel 0.
/O=*pathname*	Open the file identified by *pathname* for output on Channel 1.
/OVNODES=*n*	Limit the process to a maximum of *n* overlay nodes.
/PAGES=*n*	Limit the process to *n* pages of 1 K-words.
/PRIORITY=*n*	Set the priority for this process to *n*. This can range from 0 to 255; 0 is the highest and 255 is the lowest.
/SEGMENTS=*n*	Limit the process to a maximum of *n* attached segments, excluding segments 0, 1, and 2 (impure, shared, and overlay segments), which are automatically allocated by the system.
/TCBS=*n*	Limit the number of TCBs available to this process to a maximum of *n*.

Examples

```
) PROCESS/I=A1/O=A2/CHANNELS=10/OVNODES=5/SEGMENTS=4/TCBS=10&
&)/PRIORITY=3 MOTION.PR ↲
```

Creates a process to run concurrently with the other processes in the system. This process will execute program MOTION.PR.

PROMPT

Display or Set Prompt Sequence

) PROMPT *[command-1 [... command-4]]* ↓

Display or set the commands the CLI executes before each command prompt. Entering this command without arguments and switches displays and then executes the current prompt sequence. You can specify up to four CLI commands to be included in the sequence. Macro names or commands requiring arguments, however, cannot be used.

Initially, there are no commands in the prompt sequence.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/K	Clear all commands from the prompt setting. No arguments are allowed.
/P	Set the prompt setting to that of the previous environment level. No arguments are allowed.

Examples

```
) PROMPT ↓
```

Indicates that there are no commands in the prompt sequence. This is the default setting; each prompt will consist of only).

```
) PROMPT TIME DIRECTORY ↓
13:45:10
@DPD0:MAILBOX
)
```

Sets PROMPT to display the system time and the working directory before each command prompt, which it does before the next prompt.

PUSH Move to New CLI Environment Level

```
) PUSH ↵
```

Move CLI processing to a new environment level. The new level has the same environment values as the old level but is completely separate from it.

To return to the previous environment level and restore its values, enter the POP command (refer to POP). POP does not save the environment values as does PUSH.

Table 9.9 lists the parameters that define each CLI environment and the commands that set and display those values.

Corresponding CLI Command	Parameter
CHARACTERISTICS	Console characteristics
CLASS1	CLASS1 error response
CLASS2	CLASS2 error response
DIRECTORY	Working directory
LEVEL	Environment level
MEASURE	Elapsed time reporting mode setting
PROMPT	Prompt setting
SEARCHLIST	Searchlist
STRING	String values
TRACE	Trace mode setting

Table 9.9 The CLI environment

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/V	Verify the new environment level by displaying the level number.

Examples

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

Displays the current level, Level 0, and then PUSHes to Level 1. The CLI saves the environment for Level 0, allowing you to return to it later. Now that you are in Level 1, you can perform any operations you wish without affecting Level 0.

QPRINT

Print One or More Files

)QPRINT *pathname-1 [... pathname-n]* ⮐

Place the file(s) named in *pathname* in the printer's 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.

The command accepts pathnames, filenames, or filename templates.

A header page showing the filename, pathname, time last modified, and printing time will be printed at the beginning of the first (or only) copy of the file. With the /BANNER switch, you can add an optional text string to the information on the header page.

For more information, see the SPOOLER documentation in the *MP/AOS System Generation and Related Utilities Programmer's Reference* (DGC No. 069-400206).

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/BANNER=*text-string*	Defines optional text, up to 11 characters, to be printed on header page.
/COLUMNS=*n*	Overrides the columns-per-line setting for the duration of the files named in the command line.
/COPIES=*n*	Specifies the number of copies of each file to print. If this switch is not specified, the system prints one copy of each file.
/FOLDLINES	Continues ("wraps") lines longer than the paper width onto the next line of the page. If the switch is not specified, extra-long lines are truncated from the printed output.
/LINES=*n*	Overrides the lines-per-page setting for the duration of the files named in the command line.
/NOTIFY	Displays a message on the user's console when the file has been printed.
/QUEUE=*print-spooler*	Specifies the printer where output should be sent. If the switch is not specified, the output is printed on either @LPT or @LPB, whichever is present.

Example

```
) QPRINT/BANNER=FOR_JOANNE/COPIES=4 CIVIL ELECTRICAL MECHANICAL ↲
```

Prints four copies each of the files CIVIL, ELECTRICAL and MECHANICAL. The header page preceding each set of identical files will display the message "FOR_JOANNE" as well as the usual information about the file.

!READ Prompt for and Accept a Response (Pseudomacro)

[!READ *message*]

Displays the accompanying message, waits for you to type a response, and then uses your response as an argument to the rest of the command line.

Message is an argument string that the CLI will display as a prompt for input. Specify a message that indicates the kind of response to be entered. Your response can be a maximum of 127 characters long; complete it with a New-line or Carriage Return.

Include this pseudomacro as an argument to another command. Your response to !READ is passed to the rest of the command line.

Examples

```
) SEARCHLIST [!READ ENTER DIRECTORIES TO BE IN SEARCHLIST] ↵
ENTER DIRECTORIES TO BE IN SEARCHLIST @DPDO:UTILS,@DPDO:PROJECTS ↵
) SEARCHLIST ↵
@DPDO:UTILS,@DPDO:PROJECTS
```

Displays the !READ message as a prompt for input from the keyboard. After the response is entered, the CLI substitutes it as an argument to the SEARCHLIST command and sets the searchlist to the specified directories. The displayed searchlist indicates that it contains these directories.

RENAME

Change a File's Name or Graft It

) RENAME *current-pathname new-pathname* ⤶

Current-pathname is the current name of the file, and *new-pathname* is the name to which you want the file changed. You can do either a simple rename or a graft. A simple rename changes the filename for the file but does not change its parent directory. A graft, on the other hand, may or may not change the filename but does change the file's parent directory.

Since directories are protected by the permanence (P) attribute, you cannot rename them until you turn off P with the ATTRIBUTE command.

Only empty directories can be renamed.
RENAME does not use the searchlist.
Renaming across devices is not allowed.

Refer to Chapter 3 for more information about RENAME.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/D	Delete *new-pathname* (if it currently exists) prior to the renaming operation.

Examples

```
) RENAME FLUIDS SOLVENTS ⤶
```

Changes the name of the FLUIDS file to SOLVENTS.

```
) RENAME   @DPDO:THERMO:STATES   @DPDO:EQUILIBRIUM:STATES ⤶
```

Grafts the STATES file from directory @DPDO:THERMO to directory @DPDO:EQUILIBRIUM.

!RETURN Terminate a Macro (Pseudomacro)

[!RETURN *[argument-string]]*

Terminate the macro that is currently processing. If the macro was called as a textual substitution, !RETURN passes any arguments (or a null string in the absence of arguments) to the command line that called the macro.

Examples

```
) WRITE [!EFFICIENCY] ↲
```

(The EFFICIENCY macro contains a !RETURN)

```
        .
        .
   [!RETURN EFFICIENCY TESTS RECORDED]
```

```
EFFICIENCY TESTS RECORDED
)
```

Calls a macro named EFFICIENCY as a textual substitution. When the CLI processes !RETURN, the macro ends. The argument string is substituted for the macro call that invoked the macro and is treated as an argument to the WRITE command. WRITE displays the argument string at the console.

REVISION

Set or Display Program Revision Number

) REVISION *pathname [major.minor]* ↲

Major and *minor* are the major and minor revision numbers, which can range from 0 to 255. This enables you to mark and keep track of different versions of the same program.

NOTE: *There are two other ways of setting the revision number. You can include the .REV pseudo-op when assembling the source file, or specify the /REV= keyword switch when binding the object file.*

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/V	Displays the program name with the revision number.

Examples

```
) REVISION COUNT.PR 00.03 ↲
```

Sets the revision number of a program file called COUNT.PR.

```
) REVISION COUNT.PR ↲
00.03
```

Displays the revision number for COUNT.PR.

RUNTIME Report Process Resource Usage Statistics

) RUNTIME *[process-id]* ⤶

This command displays the following information for the process you specify:

- elapsed time since the process was created
- central processor time used
- number of blocks of data read or written
- number of characters transferred

If you do not specify a process, the statistics are reported for the CLI process issuing this command.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) RUNTIME 15 ⤶
ELAPSED 21:44:09,CPU 0:01:47.009, I/O BLOCKS 927,CHARACTERS 93901
```

Displays statistics for Process 15's use of the system resources.

```
) RUNTIME ⤶
ELAPSED 10:12:05,CPU 0:04:33.008,I/O BLOCKS 6504,CHARACTERS 45997
```

Reports resource usage statistics for the CLI.

SEARCHLIST

Set or Display Searchlist

) SEARCHLIST *[pathname-1 [... pathname-5]]* ↓

The system searches the directories in the searchlist whenever you request access to a file that cannot be found in the working directory. The directories are searched according to the order you enter them.

If you enter SEARCHLIST with arguments but no switches, the arguments replace the current searchlist. However, if you include the /A switch, the arguments are appended to the current searchlist. A maximum of five directories can be in the searchlist at any time.

The /K switch deletes the current searchlist. No arguments can be entered when you include this switch.

The /P switch enables you to change the contents of the searchlist to those of the previous environment level; again no arguments are allowed when including this switch.

To display the current searchlist, simply enter SEARCHLIST. For the root CLI, the searchlist contains the root directory of the disk from which the system was booted.

NOTE: *The searchlist is not searched when you:*

enter a fully-qualified pathname
include a prefix with the filename (e.g., = or ↑)
use the CREATE, DELETE, RENAME, or FILESTATUS commands

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/A	Add the arguments to the end of the current searchlist.
/K	Delete the current searchlist. No arguments are allowed.
/P	Change the searchlist to that of the previous environment level. No arguments are allowed.

Examples

```
) SEARCHLIST ⤶
@DPF0:
```

Displays the current searchlist.

```
) SEARCHLIST @DPF0:UTILS ⤶
) SEARCHLIST ⤶
@DPF0:UTILS
```

Changes the searchlist to the @DPF0:UTILS directory, and then displays the searchlist.

```
) SEARCHLIST/A @DPF0:MECHANIC @DPF0:ELECTRIC ⤶
) SEARCHLIST ⤶
@DPF0:UTILS,@DPF0:MECHANIC,@DPF0:ELECTRIC
```

Appends two directories:
@DPF0:MECHANIC and
@DPF0:ELECTRIC
to the current searchlist.

```
) LEVEL ⤶
LEVEL 1
) SEARCHLIST ⤶
@DPF0:UTILS,@DPF0:MECHANIC,@DPF0:ELECTRIC
) SEARCHLIST/P ⤶
) SEARCHLIST ⤶
@DPF0:UTILS
```

Displays the current level (Level 1) and the current searchlist, and then resets the searchlist to the searchlist in the previous level (Level 0). The environment level is still Level 1, but the searchlist now contains the @DPF0:UTILS directory.

```
) SEARCHLIST/K ⤶
) SEARCHLIST ⤶

)
```

Deletes the current searchlist and then checks to be sure the searchlist is in fact empty.

STRING

Set or Display the String Buffers

) STRING *[text-string]* ↲

Set or display the contents of one or more CLI string buffers. Five string buffers are available in the CLI, and each can contain up to 127 characters. You specify which buffer you want to fill by including the /S=*n* switch, where *n* is a number from 1 to 5. If you do not explicitly specify a particular buffer (with /S=*n*), the CLI writes *text-string* to String Buffer 1. *Text-string* is the information you want the string to contain. Initially, all five buffers are empty.

To clear a buffer of its contents, enter STRING with the /K switch and the /S=*n* switch, where *n* identifies the buffer you want to nullify. If you include just /K, the CLI clears only Buffer 1.

The /P switch sets the buffer you specify with /S=*n* to the contents of the corresponding string in the previous environment level. If you enter /P without /S=*n*, the contents of Buffer 1 are changed to the same contents as Buffer 1 for the previous environment level.

NOTE: *The /S switch for the EXECUTE and XEQ commands writes program termination messages only to String Buffer 1, to maintain program compatibility between MP/AOS and AOS, which has only one string buffer.*

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/K	Clear the contents of the string buffer you specify with the /S=*n* switch. When you exclude /S=*n*, this switch clears only Buffer 1. No arguments are allowed.
/P	Set the contents of the string buffer you specify with /S=*n* to the contents of the corresponding buffer in the previous environment level. Excluding /S=*n* changes only Buffer 1 to match that of the previous level. No arguments are allowed.
/S=*n*	Manipulate String Buffer *n*, where n is a number from 1 to 5.

Examples

```
) STRING ↵
BLUE
```

Displays the contents of String Buffer 1.

```
) STRING/S=3 YELLOW ↵
```

Sets String Buffer 3 to contain the text string YELLOW.

```
) STRING/K/S=4 ↵
```

Clears String Buffer 4 of its contents.

```
) LEVEL ↵
LEVEL 2
) STRING/S=(1,2) ↵
NEW
AGED
) STRING/P/S=2 ↵
) STRING/S=(1,2) ↵
NEW
OBSOLETE
```

Displays the current environment level number and the contents of String Buffers 1 and 2, and then changes String Buffer 2 to the same contents as String Buffer 2 in the previous level (Level 1). String Buffer 2 now contains the string OBSOLETE; the other string is unaffected.

SWAPLEVEL

Display CLI Program Swap Level

```
) SWAPLEVEL ↲
```

Display the program swap level at which the CLI is executing. The level number that is displayed corresponds to the CLI's level in the process' program stack.

Level 1 is the first program swap level.

Do not confuse SWAPLEVEL with LEVEL, which returns the current CLI environment level number.

Refer to Chapter 5, "Program and Process Management," and the *MP/AOS System Programmer's Reference* (DGC No. 093-400051) for more information about program swap levels and swap and chain operations.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) SWAPLEVEL ↲
LEVEL 1
```

The CLI is executing at the first program swap level for this process.

```
) SWAPLEVEL ↲
LEVEL 1
) XEQ MASM LOG_M ↲
) SWAPLEVEL ↲
LEVEL 1
```

SYSTEM

Display MP/AOS Status

) SYSTEM ⤶

This command displays the following information:

- system revision number
- amount of physical memory used by the system
- number of pages free in memory
- number of concurrent processes the system can support

Including the /ID or /DISK switch displays more information about the system. /ID displays the system identification, and /DISK displays the name of the disk from which the system was booted.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/DISK	Display the name of the disk from which the system was booted.
/ID	Display the system identification.

Examples

```
) SYSTEM ↲
System Information
   Revision: 1.00
   Physical memory pages in use: 115
   Pages available: 141
   Maximum concurrent processes: 8
```

Displays the status of the current operating system.

```
) SYSTEM/DISK ↲
System Information
   Revision: 1.00
   Physical memory pages in use: 115
   Pages available: 141
   Maximum concurent processes: 8
   System disk: @DPD0
```

Displays the status of the current operating system. MP/AOS was booted from @DPD0.

```
) SYSTEM/ID ↲
System Information
   Revision: 1.00
   Physical memory pages in use: 115
   Pages available: 141
   Maximum concurrent processes: 8
   System ID: MYSYS
```

The ID of the currently running system is MYSYS.

TERMINATE End a Process

) TERMINATE *process-id* ↵

Terminate the specified process.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/BREAK	Create a break file for the specified process.

Examples

```
) TERMINATE 6 ↵
```

Ends Process 6.

```
) TERMINATE/BREAK 5 ↵
```

Ends Process 5 after creating a break file to save its current processing values.

TIME

Set or Display System Time

) TIME *[hh:mm:ss]* ↓

Enter the time as *hh* for hours, *mm* for minutes, and *ss* for seconds. If you enter TIME without an argument, the CLI displays the current system time.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) TIME 15:22:19 ↓
```

Sets the system time.

```
) TIME ↓
15:22:23
```

Displays the current system time.

TRACE Set or Display Command Trace Mode

) TRACE *[ON | OFF]* ↵

Turn the command line trace mode on or off, or display its current setting.

By default the trace mode is off. If you enter TRACE without an argument, the CLI displays the current trace mode setting.

When Trace mode is on, CLI commands are preceded with ***, textual substitutions and pseudomacros with !!!, and macro calls with ###.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) TRACE ↵
OFF
) TRACE ON ↵
```

Displays the current trace mode setting, and then turns on trace mode.

The macro NOW.CLI:

```
  WRITE IT IS NOW [!TIME] ON [!DATE]
```

produce the following trace output:

```
NOW
### [NOW]
!!! [TIME],ON,[!DATE]
!!! [!DATE]
*** WRITE,IT,IS,NOW,22:31:46:ON,04-MAR-82

IT IS NOW 22:31:46 ON 04-MAR-82
```

TYPE

Display File Contents

) TYPE *pathname-1 [...pathname-n]* ⤶

This command accepts templates in place of pathnames.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/V	Display the name of each file before typing it.

Example

```
) TYPE FILEXYZ ⤶
```

Displays the contents of FILEXYZ on the console.

UADD Sum Two Decimal Integers

) UADD *integer1 integer2* ↵

Compute and display the sum of two decimal integers. Both numbers you add must be unsigned, decimal integers in the range 0 to 4,294,967,295.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) UADD 255 690 ↵
945
```

Calculates the sum of 255 and 690.

UDIVIDE

Divide Unsigned Integers

) UDIVIDE *dividend divisor* ⤶

Divide one unsigned integer by another, and list the resulting integer quotient. The CLI divides the *dividend* by the *divisor*. The *dividend* must be an unsigned integer in the range 0 to 4,294,967,295, whereas the *divisor* is an unsigned integer in the range 1 to 65,535. Remainders are discarded, so the result is an integer.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) UDIVIDE 255 15 ⤶
17
```

Divides 255 by 15, and lists the resulting integer quotient.

```
) UDIVIDE 1 5 ⤶
0
```

Calculates the integer quotient for 1 divided by 5. Notice that the remainder is discarded.

UMODULO Perform a Modulo Operation

) UMODULO *integer modulus* ⤶

Compute and display the result of *integer* modulo *modulus. Integer* is in the range 0 to 4,294,967,295. *Modulus* can range from 0 to 65,535.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) UMOD 12 8 ⤶
4
```

Computes 12 to modulo 8. The result is 4.

UMULTIPLY

Multiply Unsigned Integers

) UMULTIPLY *integer1 integer2* ↵

Compute and display the product of two unsigned decimal integers.

Both arguments are unsigned decimal integers. *Integer1* is in the range 0 to 4,294,967,295; *integer2* is in the range 0 to 65,535.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) UMUL 300 12 ↵
3600
```

Multiplies 300 by 12 and displays the product, 3600.

UNBLOCK Allow a Blocked Process to Run

) UNBLOCK *process-id* ⤸

Allow a blocked process to resume processing. If the process you specify is not blocked, this command will have no effect on the system. Chapter 5 describes blocking and unblocking processes.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) BLOCK 7 ⤸
) UNBLOCK 7 ⤸
```

Block Process 7 from further processing, and then unblock the process so it will resume execution. A blocked process will remain blocked until another process unblocks it.

USUBTRACT

Subtract Unsigned Integers

) USUBTRACT *integer1 integer2* ↲

Subtract one unsigned, decimal integer *(integer2)* from another *(integer1)* and display the result. Both integers must be in the range 0 to 4,294,967,295. The result will be an unsigned, decimal integer in the same range.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) USUBTRACT 350 119 ↲
231
```

Subtracts 119 from 350 and displays the result, 231.

```
)USUBTRACT 3 12 ↲
4294967287
```

Subtracts 12 from 3. The result is a negative number, which "wraps around" to give a very large unsigned number, 9 integers from the top of the range.

WHO Identify a Process

) WHO *[process-id]* ⤶

Identify the specified process and indicate what it is doing.

This command displays the process identification number (PID), and the name of the program the process is currently executing. If you do not enter a PID number, the information displayed is about the CLI. Process management is explained in Chapter 5.

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Examples

```
) WHO 7 ⤶
PID: 7 @DPD0:UTIL:MASM.PR
```

Process 7 is executing the Macroassembler.

```
) WHO ⤶
PID: 9 @DPH0:CLI.PR
```

Indicates that the current process is PID 9, and that it is executing the CLI.

WRITE

Display Argument(s)

) WRITE *[argument-1 ...argument-n]* ⤸

This command writes the arguments you specify to the output device exactly as you enter them.

CLI Standard Command Switches

Switch	Action
/1 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2 =*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L =*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

None.

Example

```
) WRITE LIST THE RESOURCES ⤸
LIST THE RESOURCES
```

Displays the arguments as entered.

XEQ Swap in a New Program

) XEQ *program-name [program-arguments]* ⤶

Identical to EXECUTE. XEQ writes the CLI out to disk and reads a program into memory to run in its place. Any arguments you enter are passed to the specified program. MP/AOS swaps out the CLI and saves its processing status on disk. The program swap level is incremented by one, and the specified program executes in the same process address space as the CLI, but at the incremented level. (You can nest program levels to Level 8.)

If XEQ is issued with the /M switch from within a macro, the new program accepts as input the macro lines following the XEQ command line. The last input line must be a single).

Another way to pass input to the program is to include the /I=*pathname* switch, where *pathname* is an existing script file. A script file contains instructions to be entered in place of console input to achieve specific results without keyboard intervention.

To write program output to a file instead of to the console, include the /O=*pathname* switch.

If you include the /S switch, the program termination message is written to String Buffer 1 instead of to the console.

When processing of the program ends, control is returned to the parent program, the CLI, which executes at its original program swap level. The CLI will report any exception conditions returned by the executed program or by the MP/AOS system. Refer to Chapter 5 for further information about program execution.

NOTE: *XEQ functions identically to EXECUTE.*

CLI Standard Command Switches

Switch	Action
/1=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS1 exception condition that occurs while this command is executing.
/2=*error-response*	Set the error response to IGNORE, WARNING, or ERROR for any CLASS2 exception condition that occurs while this command is executing.
/L	Write output to the line printer instead of to the default output device.
/L=*pathname*	Write CLI output to the file specified by *pathname* instead of to the default output device.
/TIMEMODE	Report the time used to execute this command.

Command Specific Switches

Switch	Action
/I=*pathname*	Reads program input from the specified file instead of from the console.
/M	Read program input from the currently executing macro instead of from the console. The last input line must be a single).
/O=*pathname*	Writes the program output to the specified file instead of to the console. If you give a simple filename that does not already exist, it will be created.
/S	Writes the program termination message to String Buffer 1 instead of to the console. This switch cannot be used with /L(=).

Examples

```
) XEQ PROGRAMX ↲
```

Swaps out the CLI and then executes PROGRAMX in its place. MP/AOS saves the processing status of the CLI on disk.

```
) XEQ MASM PROGY ↲
```

Invokes the Macroassembler, passing it the name PROGY, the source file to be assembled.

```
) XEQ/S CONSTRUCT.PR ↲
```

Executes program CONSTRUCT.PR. The program termination message will be written to String Buffer 1 for later use.

```
) XEQ/I=RESOURCES/O=TIMETABLE PROJECT_PLAN.PR ↲
```

Invokes the program named PROJECT_PLAN.PR, using the RESOURCES file for input. The program will write its output to TIMETABLE, rather than to the console.

CLI Error Codes A

Error Code	Mnemonic	Message
41001	ECUSW	Unknown switch specified
41002	ECMRS	Command requires switch
41003	ECCSW	Conflicting switches specified
41004	ECMRA	Command requires argument(s)
41005	ECTMA	Too many arguments specified
41006	ECINA	Improper argument given for command
41007	ECTFA	Too few arguments specified
41010	ECUKC	Unknown command, macro or program name
41011	ECAMC	Ambiguous command abbreviation
41013	ECUOB	Unmatched or unbalanced <74>
41014	ECUCB	Unmatched or unbalanced <76>
41015	ECUOP	Unmatched or unbalanced (
41016	ECUCP	Unmatched or unbalanced)
41017	ECUOS	Unmatched or unbalanced [
41020	ECUCS	Unmatched or unbalanced]
41021	ECAMS	Ambiguous switch abbreviation
41022	ECINS	Invalid switch value
41023	ECSRV	Switch must be a keyword switch
41024	ECSNV	Switch is not a keyword switch
41025	ECPMS	Pathname must start in current directory
41026	ECNFM	No filenames match template
41027	ECMMD	Missing macro input delimiter
41030	ECZLF	Zero-length filename specified
41031	ECNMA	Insufficient memory available for operation
41052	ECUNC	Unimplemented command
41053	ECNPO	Cannot pop from level 0
41054	ECNFO	Operation not allowed from level 0
41055	ECNSW	No spooler available
41056	ECMBM	Operation must occur within macro
41057	ECNOP	No open pseudo-macro block
41060	ECOCA	Only commands allowed
41061	ECMDQ	Missing literal closing quotes

Table A.1 CLI error messages

The ASCII Character Set

B

DECIMAL	OCTAL	HEX	KEY SYMBOL	MNEMONIC
0	000	00	↑@	NUL
1	001	01	↑A	SOH
2	002	02	↑B	STX
3	003	03	↑C	ETX
4	004	04	↑D	EOT
5	005	05	↑E	ENQ
6	006	06	↑F	ACK
7	007	07	↑G	BEL
8	010	08	↑H	BS (BACKSPACE)
9	011	09	↑I	TAB
10	012	0A	↑J	NEW LINE
11	013	0B	↑K	VT (VERT. TAB)
12	014	0C	↑L	FORM FEED
13	015	0D	↑M	CARRIAGE RETURN
14	016	0E	↑N	SO
15	017	0F	↑O	SI
16	020	10	↑P	DLE
17	021	11	↑Q	DC1
18	022	12	↑R	DC2
19	023	13	↑S	DC3
20	024	14	↑T	DC4
21	025	15	↑U	NAK
22	026	16	↑V	SYN
23	027	17	↑W	ETB
24	030	18	↑X	CAN
25	031	19	↑Y	EM
26	032	1A	↑Z	SUB
27	033	1B	ESC	ESCAPE
28	034	1C	↑\	FS
29	035	1D	↑]	GS
30	036	1E	↑↑	RS
31	037	1F	↑←	US

DECIMAL	OCTAL	HEX	KEY SYMBOL
32	040	20	SPACE
33	041	21	!
34	042	22	" (QUOTE)
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	' (APOS)
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	, (COMMA)
45	055	2D	-
46	056	2E	. (PERIOD)
47	057	2F	/
48	060	30	0
49	061	31	1
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	<
63	077	3F	?
64	100	40	@

DECIMAL	OCTAL	HEX	KEY SYMBOL
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	↑ OR ∧
95	137	5F	← OR —
96	140	60	` (GRAVE)

DECIMAL	OCTAL	HEX	KEY SYMBOL
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~ (TILDE)
127	177	7F	DEL (RUBOUT)

DG-05495

I/O Device Mnemonics

C

Unit Name	ECLIPSE	microNOVA
DPG	20 Mbyte cartridge disk	—
DPD	10 Mbyte cartridge disk 315 Kbyte diskette	10 Mbyte cartridge disk
DPH	12.5/25 Mbyte fixed disk 1.25 Mbyte diskette	12.5/25 Mbyte fixed disk 1.25 Mbyte diskette
DPF	50/95/190 Mbyte cartridge disk	—
DPX	—	315 Kbyte diskette
TTI	Console keyboard (input)	Console keyboard (input)
TTO	Console display (output)	Console display (output)
LPB	Data channel line printer	Data channel line printer
LPT	Programmed I/O line printer	Programmed I/O line printer
LP2	Data channel (or programmed I/O) line printer	Data channel (or programmed I/O) line printer

Table C.1 Disk(ette), console, line printer unit names

Index

DG OFFICES

NORTH AMERICAN OFFICES

Alabama: Birmingham
Arizona: Phoenix, Tucson
Arkansas: Little Rock
California: Anaheim, El Segundo, Fresno, Los Angeles, Oakland, Palo Alto, Riverside, Sacramento, San Diego, San Francisco, Santa Barbara, Sunnyvale, Van Nuys
Colorado: Colorado Springs, Denver
Connecticut: North Branford, Norwalk
Florida: Ft. Lauderdale, Orlando, Tampa
Georgia: Norcross
Idaho: Boise
Iowa: Bettendorf, Des Moines
Illinois: Arlington Heights, Champaign, Chicago, Peoria, Rockford
Indiana: Indianapolis
Kentucky: Louisville
Louisiana: Baton Rouge, Metairie
Maine: Portland, Westbrook
Maryland: Baltimore
Massachusetts: Cambridge, Framingham, Southboro, Waltham, Wellesley, Westboro, West Springfield, Worcester
Michigan: Grand Rapids, Southfield
Minnesota: Richfield
Missouri: Creve Coeur, Kansas City
Mississippi: Jackson
Montana: Billings
Nebraska: Omaha
Nevada: Reno
New Hampshire: Bedford, Portsmouth
New Jersey: Cherry Hill, Somerset, Wayne
New Mexico: Albuquerque
New York: Buffalo, Lake Success, Latham, Liverpool, Melville, New York City, Rochester, White Plains
North Carolina: Charlotte, Greensboro, Greenville, Raleigh, Research Triangle Park
Ohio: Brooklyn Heights, Cincinnati, Columbus, Dayton
Oklahoma: Oklahoma City, Tulsa
Oregon: Lake Oswego
Pennsylvania: Blue Bell, Lancaster, Philadelphia, Pittsburgh
Rhode Island: Providence
South Carolina: Columbia
Tennessee: Knoxville, Memphis, Nashville
Texas: Austin, Dallas, El Paso, Ft. Worth, Houston, San Antonio
Utah: Salt Lake City
Virginia: McLean, Norfolk, Richmond, Salem
Washington: Bellevue, Richland, Spokane
West Virginia: Charleston
Wisconsin: Brookfield, Grand Chute, Madison

DG-04976

INTERNATIONAL OFFICES

Argentina: Buenos Aires
Australia: Adelaide, Brisbane, Hobart, Melbourne, Newcastle, Perth, Sydney
Austria: Vienna
Belgium: Brussels
Bolivia: La Paz
Brazil: Sao Paulo
Canada: Calgary, Edmonton, Montreal, Ottawa, Quebec, Toronto, Vancouver, Winnipeg
Chile: Santiago
Columbia: Bogata
Costa Rica: San Jose
Denmark: Copenhagen
Ecuador: Quito
Egypt: Cairo
Finland: Helsinki
France: Le Plessis-Robinson, Lille, Lyon, Nantes, Paris, Saint Denis, Strasbourg
Guatemala: Guatemala City
Hong Kong
India: Bombay
Indonesia: Jakarta, Pusat
Ireland: Dublin
Israel: Tel Aviv
Italy: Bologna, Florence, Milan, Padua, Rome, Tourin
Japan: Fukuoka, Hiroshima, Nagoya, Osaka, Tokyo, Tsukuba
Jordan: Amman
Korea: Seoul
Kuwait: Kuwait
Lebanon: Beirut
Malaysia: Kuala Lumpur
Mexico: Mexico City, Monterrey
Morocco: Casablanca
The Netherlands: Amsterdam, Rijswijk
New Zealand: Auckland, Wellington
Nicaragua: Managua
Nigeria: Ibadan, Lagos
Norway: Oslo
Paraguay: Asuncion
Peru: Lima
Philippine Islands: Manila
Portugal: Lisbon
Puerto Rico: Hato Rey
Saudi Arabia: Jeddah, Riyadh
Singapore
South Africa: Cape Town, Durban, Johannesburg, Pretoria
Spain: Barcelona, Bibao, Madrid
Sweden: Gothenburg, Malmo, Stockholm
Switzerland: Lausanne, Zurich
Taiwan: Taipei
Thailand: Bangkok
Turkey: Ankara
United Kingdom: Birmingham, Bristol, Glasgow, Hounslow London, Manchester
Uruguay: Montevideo
USSR: Espoo
Venezuela: Maracaibo
West Germany: Dusseldorf, Frankfurt, Hamburg, Hannover, Munich, Nuremburg, Stuttgart

Data General

Ordering Technical Publications

TIPS is the Technical Information and Publications Service—a new support system for DGC customers that makes ordering technical manuals simple and fast. Simple, because TIPS is a central supplier of literature about DGC products. And fast, because TIPS specializes in handling publications.

TIPS was designed by DG's Educational Services people to follow through on your order as soon as it's received. To offer discounts on bulk orders. To let you choose the method of shipment you prefer. And to deliver within a schedule you can live with.

How to Get in Touch with TIPS

Contact your local DGC education center for brochures, prices, and order forms. Or get in touch with a TIPS administrator directly by calling (617) 366-8911, extension 4086, or writing to

Data General Corporation
Attn: Educational Services, TIPS Administrator
MS F019
4400 Computer Drive
Westborough, MA 01580

TIPS. For the technical manuals you need, when you need them.

DGC Education Centers

Boston Education Center
Route 9
Southboro, Massachusetts 01772
(617) 485-7270

Washington, D.C. Education Center
7927 Jones Branch Drive, Suite 200
McLean, Virginia 22102
(703) 827-9666

Atlanta Education Center
6855 Jimmy Carter Boulevard, Suite 1790
Norcross, Georgia 30071
(404) 448-9224

Los Angeles Education Center
5250 West Century Boulevard
Los Angeles, California 90045
(213) 670-4011

Chicago Education Center
703 West Algonquin Road
Arlington Heights, Illinois 60005
(312) 364-3045

Data General

Technical Products Publications Comment Form

Please help us improve our future publications by answering the questions below. Use the space provided for your comments.

Title: ____________________

Document No. 069-400201-00

Yes	No			
☐	☐	Is this manual easy to read?	○ You (can, cannot) find things easily. ○ Language (is, is not) appropriate. ○ Technical terms (are, are not) defined as needed.	○ Other:
		In what ways do you find this manual useful?	○ Learning to use the equipment ○ As a reference ○ As an introduction to the product	○ To instruct a class. ○ Other:
☐	☐	Do the illustrations help you?	○ Visuals (are,are not) well designed. ○ Labels and captions (are,are not) clear. ○ Other:	
☐	☐	Does the manual tell you all you need to know? What additional information would you like?		
☐	☐	Is the information accurate? (If not please specify with page number and paragraph.)		

Name: __________ Title: __________

Company: __________ Division: __________

Address: __________ City: __________

State: ______ Zip: ______ Telephone: ______ Date: ______

DG-06895

Data General

Data General Corporation, Westboro, Massachusetts 01580

CUT ALONG DOTTED LINE

FOLD FOLD

TAPE 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: Technical Products Publications (C-138)
4400 Computer Drive
Westboro, MA 01581

Data General

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?

○ ______________

____

Data General

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

CUT ALONG DOTTED LINE

FOLD FOLD

TAPE 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

Data General Corporation, Westboro, Massachusetts 01580

069-4002