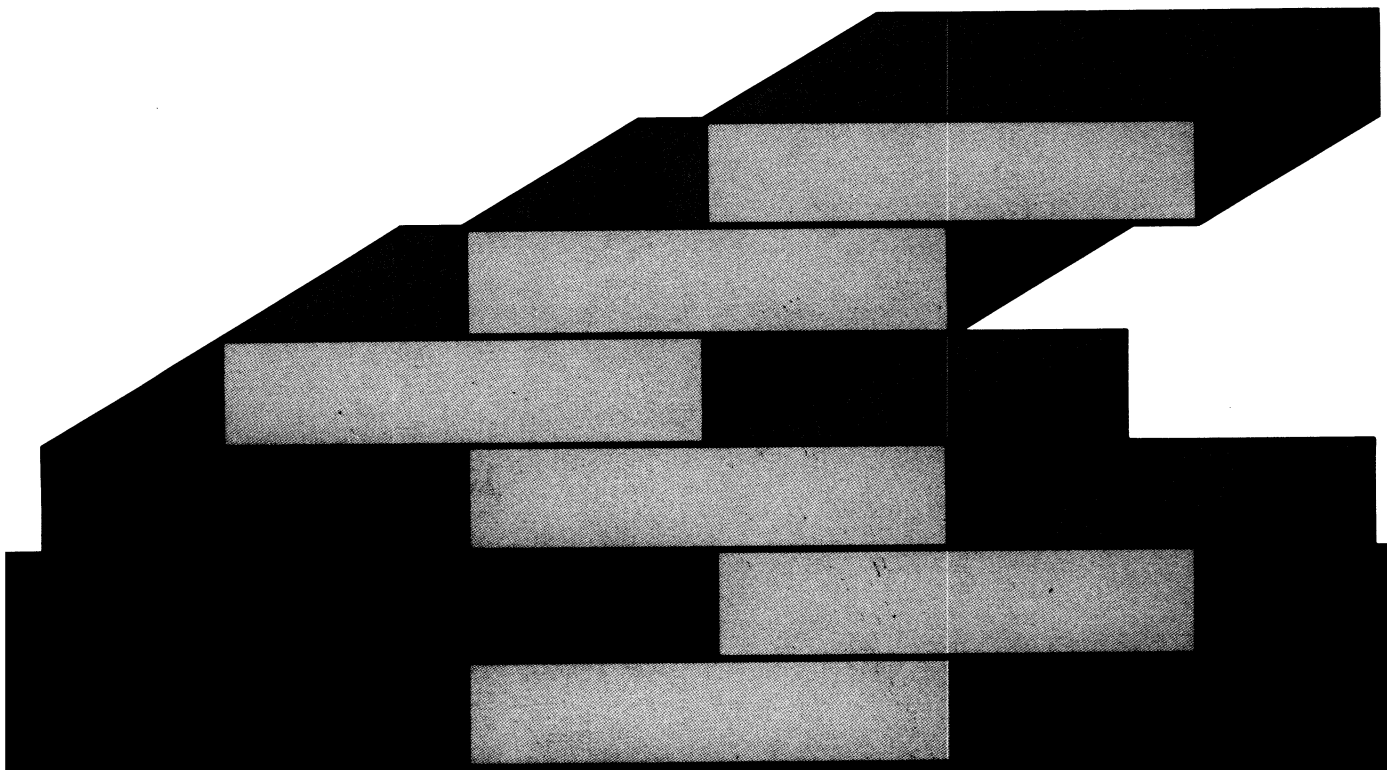
AOS/RT32
Command Line Interpreter



Data General

AOS/RT32

Command Line Interpreter

Notice

Data General Corporation (DGC) has prepared this document for use by DGC personnel, customers, and prospective customers. The information contained herein shall not be reproduced in whole or in part without DGC's prior written approval.

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

The terms and conditions governing the sale of DGC hardware products and the licensing of DGC software consist solely of those set forth in the written contracts between DGC and its customers. No representation or other affirmation of fact contained in this document including but not limited to statements regarding capacity, response-time performance, suitability for use or performance of products described herein shall be deemed to be a warranty by DGC for any purpose, or give rise to any liability of DGC whatsoever.

In no event shall DGC be liable for any incidental, indirect, special or consequential damages whatsoever (including but not limited to lost profits) arising out of or related to this document or the information contained in it, even if DGC has been advised, knew or should have known of the possibility of such damages.

NOVA, INFOS, ECLIPSE, ECLIPSE MV/8000, DASHER, microNOVA, ENTERPRISE, and **PROXI** are U.S. registered trademarks of Data General Corporation, and **AZ-TEXT, DASHER, DG/L, ECLIPSE MV/6000, METEOR, PRESENT, REV-UP, SWAT, XODIAC, GENAP,** and **TRENDVIEW** are U.S. trademarks of Data General Corporation.

Ordering No. 069-400028

Printed in the United States of America
Rev. 00, February 1982

Preface

This guide was written for computer operators and system programmers. It describes the features and commands of the AOS/RT32 CLI, providing users with sufficient information to monitor and modify an AOS/RT32 environment. The guide assumes a working knowledge of the AOS/VS CLI and places special emphasis on orienting users to the more compact CLI supported by AOS/RT32.

Organization

The contents include five chapters, three appendices, and an index. Chapter 1 presents the differences between the AOS/RT32 and AOS/VS CLIs in general terms. These differences are highlighted by category at the end of each subsequent chapter.

Chapter 1, "The AOS/RT32 CLI," introduces the AOS/RT32 operating system and CLI; explains how to generate the CLI; discusses how the CLI manipulates files, processes, and devices; and summarizes substantial reductions in other areas of CLI control.

Chapter 2, "Operating Guidelines," presents the terminal features and control characters that apply to the AOS/RT32 CLI; demonstrates the use of command line syntax and coding aids; and explains the system of error message handling.

Chapter 3, "File and Macro Facilities," covers file naming and referencing; discusses generic filenames and device names; describes the filestatus, copy, and display features; and explains the use of macros under AOS/RT32.

Chapter 4, "Process and Device Control," explains how processes are identified and referred to under AOS/RT32; presents the CLI commands for process monitoring, scheduling, termination, and communication; and discusses the CLI commands that apply to devices as physical entities.

Chapter 5, "CLI Command Dictionary," presents a summary of all CLI commands in table form; and provides an alphabetically-arranged description of each CLI command that includes a sample format, table of applicable switches, and examples of valid command usage.

Appendix A defines the exceptional condition messages specific to the AOS/RT32 CLI.

Appendix B lists the system's error codes according to their numeric, assembled values.

Appendix C contains the ASCII character set.

Related Manuals

If you are not an experienced assembly language programmer, study the following manuals before reading this guide.

- *Fundamentals of Small Computer Programming* (DGC No. 093-000090), a general introduction to Data General computers.
- *ECLIPSE MV/8000 Principles of Operation* (DGC No. 014-000648), a description of the MV/8000 hardware and its assembly language instruction set.
- *Advanced Operating System/Virtual Storage (AOS/VS) Macroassembler (MASM) Reference Manual* (DGC No. 093-000242), a reference guide to the syntax of AOS/VS assembly language and the Macroassembler utility.

This guide is one in a series of documents, listed below, that support the AOS/RT32 operating system.

- *AOS/RT32 System Generation* (DGC No. 069-400027) explains how to generate an AOS/RT32 system tailored to your needs.
- *AOS/RT32 Debugger* (DGC No. 069-400029) discusses the procedures for monitoring program execution and for detecting and eliminating errors.
- *AOS/RT32 System Programmer's Reference* (DGC No. 093-400016) describes the AOS/RT32 functions needed to code a program, as well as the AOS/RT32 system calls that facilitate the coding process.

Because the AOS/RT32 operating system is a subset of the Advanced Operating System/Virtual Storage (AOS/VS) operating system, you may find the following AOS/VS manuals helpful.

- *Command Line Interpreter User's Manual* (AOS/VS) (DGC No. 092-000122)
- *SPEED Text Editor User's Manual* (AOS/VS) (DGC No. 093-000197)

- *AOS/VS Programmer's Manual* (DGC No. 093-000241)
- *AOS/VS Macroassembler Manual* (DGC No. 093-000242)
- *Managing AOS/VS* (DGC No. 093-000243)
- *AOS/VS Operator's Guide* (DGC No. 093-000244)
- *AOS/VS Link and Editor User's Manual* (DGC N093-000245)
- *AOS/VS Debugger User's Manual* (DGC No. 093-000246)
- *AOS/VS SED Text Editor User's Manual* (DGC No. 093-000249)

Conventions

))	This is the AOS/RT32 CLI prompt. It indicates that the CLI is ready to accept commands.
COMMAND	This typeface is used for command mnemonics, as in COPY and TERMINATE, when they appear in examples. Textual references to commands appear in uppercase Roman type.
argument	This statement, in lowercase italic, signifies that a command argument is required. In your program, you must replace this symbolic phrase with a specific argument, such as file1.
[argument] *[/switch]*	These statements, in lowercase italic and surrounded by brackets, represent an optional argument or command switch. Do not include the brackets when you type in optional arguments or switches; they only indicate a choice.
arg1 / arg2	This typeface, in lowercase italic with a vertical bar (|), denotes that you have a choice between arg1 and arg2; that is, you may use either a filename or a device as the argument.
↓	This sign represents a new line.

In all examples, the material to be typed by the user appears as

```
)) COMMAND ARGUMENT
```

The CLI's response appears as

RESPONSE

Table of Contents

Chapter 1

The AOS/RT32 CLI

This chapter orients readers to the use of a more compact CLI than the one they are accustomed to working with. First the AOS/RT32 operating system and CLI are introduced. Then instructions for generating the AOS/RT32 CLI are provided. A general discussion of the differences between the AOS/RT32 and AOS/VS command line interpreters follows. The chapter concludes by explaining substantial reductions in CLI features where system management, environmental control, and queue facilities are concerned. As mentioned in the Preface, a prior reading of the *AOS/RT32 System Programmer's Reference* and *AOS/RT32 System Generation Manual* is strongly recommended.

The AOS/RT32 Operating System

A subset of the AOS/VS (Advanced Operating System/Virtual Storage) operating system, AOS/RT32 was designed for upward compatibility while serving real-time, rather than general, applications. It is small, fast, extremely reliable, and can be configured to your application's exact needs.

Although it supports many AOS/VS features, AOS/RT32 is not merely a smaller version of AOS/VS. Many AOS/VS features were changed internally to provide AOS/RT32 with maximum real-time response. The result is an operating system of greater sophistication and economy—a system whose 32-bit addressing capability, multi-processing support, and compatibility with existing Data General software make it an outstanding performer of scientific and industrial tasks.

The AOS/RT32 CLI

Like all command line interpreters, the AOS/RT32 CLI is a communicative link to the computer—an interactive programming language whose built-in commands allow you to monitor and modify the environment of your operating system. The capabilities of a CLI depend on the diversity of the environment it must negotiate. You are already familiar with the general-purpose AOS/VS operating system and the many features that its interpreter, the AOS/VS CLI, supports. In contrast, the AOS/RT32 CLI is designed to facilitate your interaction with a more highly focused environment—one that you created and configured during system generation to meet a specific need. Largely predetermined and relatively fixed, this environment can be controlled with the most streamlined of command line interpreters. This is the AOS/RT32 CLI.

Generating the AOS/RT32 CLI

Like all other processes that you intend to run under AOS/RT32, the CLI must be defined during the Process Define phase of system generation. Both the AOS/RT32 CLI and Process Define programs are contained in the utilities directory of the AOS/RT32 release tape. Be sure to include this directory in your searchlist while under AOS/VS and before generating the CLI. Then, while running the AOS/RT32 Process Define program, supply the following information.

- Designate the process name as CLI.PR.
- Set the number of concurrent system calls to 2.
- Assign the CLI's priority in relation to other processes in the environment.
- Specify a maximum memory size of at least 40 pages.
- Set @INPUT and @OUTPUT to the desired input and output devices for the CLI, for example, @CON2.

Differences Between the AOS/RT32 and AOS/VS CLIs

Two factors explain many of the differences between the AOS/RT32 and AOS/VS CLIs. First, the processes and files that compose an AOS/RT32 environment are created under AOS/VS, defined during system generation, and remain more or less fixed thereafter. Second, such functions as the scaling of a multi-leveled directory tree are superfluous to the management of a "flat" AOS/RT32 environment. Thus, of the 68 commands performed by the AOS/VS CLI, the AOS/RT32 command line interpreter performs 18.

The following comparison highlights the differences between the two CLIs as it explains, in general terms, the

CLI features available under AOS/RT32. These features are presented in three sections: (1) manipulating files, (2) manipulating processes, and (3) manipulating devices. They are the areas most essential to your work with the AOS/RT32 CLI.

Manipulating Files

Under AOS/RT32, a *file* refers to a collection of related data treated as a unit. All AOS/RT32 files are defined during the File Define phase of the system generation procedure; the AOS/RT32 CLI cannot create or delete them. Instead, AOS/RT32 supports uninitialized files. These are "potential" files that contain no information; they have only a name, a current size, and a predefined maximum size.

An uninitialized file provides a storage area for data external to user programs. Information can be transferred to and from it; its contents can fluctuate from zero to as many bytes of data as its maximum size permits. Meanwhile, all unused memory space is reallocated by the system, ensuring the most productive use of memory resources.

Under AOS/RT32, all file information is stored in memory rather than on disk. Thus, the CLI requires no "path" other than a name to locate any file. It monitors the status of files singly rather than in directory-related groups. The copy capability does not create a file where none exists or expand a file beyond its predefined size. And a modified macro facility allows users to define a macro file during system generation only. The CLI executes the commands it contains when they are called on with the macro name.

Chapter 3 treats these subjects in more detail and explains how to use the file and macro facilities of the AOS/RT32 CLI.

Manipulating Processes

A process is a program file residing in memory. It contains user code created under AOS/VS. During system generation, this code is combined with information that defines and controls the real-time resource management performed by AOS/RT32 for its users. Thus, a process also represents facilities allocated to the executing program, including constraints on that program.

Unlike the AOS/VS CLI, the AOS/RT32 Command Line Interpreter does not initiate processes or define their environments. There is no process hierarchy — no parents and descendants, no superior or inferior processes, no family tree. Instead, the AOS/RT32 CLI acts as a kind of window onto a set number of precreated processes, allowing users to view them individually or as a group, to schedule them as needed, and to initiate or receive console messages.

Chapter 4 explains these functions of the AOS/RT32 CLI in detail.

Manipulating Devices

AOS/RT32 supports as many as 64 devices per system, including a complete range of disk drives, magnetic tape drives, CRT and hard-copy terminals, and line printers. Some of these devices are supported for block I/O, others for record I/O. All devices must be defined and enabled during the system generation procedure.

The AOS/RT32 CLI allows users to transfer information from one device to another, from a file to a device, and from a device to a file. It provides commands for setting or displaying the features of character devices, and for controlling the rewind mechanism of a tape unit. Operator commands—commands that request an operator's assistance with a device—are not supported, however.

Chapter 3 explains how to reference devices and copy data to and from them in the AOS/RT32 CLI, while Chapter 4 describes the CLI functions that govern devices as physical entities.

Additional CLI Reductions

System management is not required of the AOS/RT32 CLI because so many of the system's variables are fixed at the time of its generation. No system manager or operator with special privileges is assumed. Instead, all users are Superusers under AOS/RT32; the CLI's only constraint on users is its Superprocess Mode, which all processes can control. The system does, however, maintain a clock and calendar, which can be set or displayed from the CLI.

AOS/RT32 does not support the following AOS/VS facilities for system management:

Superuser Mode
control messages
log file maintenance
grafting of logical disks
display of CPU identification number
setting or display of system identifier

Environmental control under AOS/VS enables a user to manipulate the parameters of his CLI process. These parameters determine the contents of the prompt and string buffers, the setting of one's default ACL (Access Control List), the CLI's response to an exceptional condition, and other factors. Ordinarily, CLI parameters under AOS/VS are first set by a system manager and later altered by a user as the need arises.

Many of the CLI parameters that apply to AOS/VS do not apply to AOS/RT32. And those that do are fixed, for the most part, during the process of system generation. Only one environmental factor can be controlled from the AOS/RT32 CLI: Superprocess Mode. The parameters not supported by the AOS/RT32 CLI include:

directory
searchlist
default ACL
listfile
datafile
prompt buffer
string buffer
Trace Mode
Squeeze Mode
Screenedit Mode
Superuser Mode
environment level
exceptional condition setting

Queue facilities help control the flow of traffic between input and output devices in a time-sharing environment. The highly structured nature of an AOS/RT32 environment obviates the need for these facilities. Thus, under AOS/RT32, the CLI COPY command supplants the use of all queue-related commands. More specifically, AOS/RT32 does not support the following queue facilities or CLI commands associated with them:

plotter queues
line printer queues
paper tape punch queues
batch or spooler queues

Chapter 2

Operating Guidelines

This chapter begins with a description of the uses of special terminal keys, including control characters and sequences. It then explains command line syntax and error message handling. A summary of operational differences between the AOS/RT32 and AOS/VS CLIs concludes the chapter.

Special Terminal Keys

Your DASHER® video display or hard-copy terminal has a keyboard with alphanumeric characters and function keys. Many of the function keys apply to utilities; they serve no special purpose for the AOS/RT32 CLI. Others, presented in Table 2.1, perform special operations. For details on a particular function key, consult the *AOS/VS Command Line Interpreter User's Manual* (DGC No. 093-000122-04).

Function Key	Effect
DEL or RUBOUT	On video display terminals, erases the character preceding the cursor and positions the cursor in its place. On hard-copy terminals, echos a backarrow or underscore, indicating that the previous character has been deleted.
NEW LINE and CR	Both keys function as terminators. Typed after a command line, either key signals the CLI to execute that command.
ERASE PAGE	On video display terminals, clears the display area and moves cursor to the top, left-hand corner of the screen. Also acts as a command terminator. (If Page Mode characteristic is on, use of this key may freeze the terminal; type CTRL-Q to free it.)
REPEAT	Depressed with one or more alphanumeric or punctuation keys, causes repetition of the generated code at a rate of 20 characters per second. (On D200 terminals, an alphanumeric or punctuation key pressed longer than one second causes its repetition at a rate of eight characters per second.)
SHIFT	Prints alphabetic characters in uppercase, and prints the shift characters for all other keys.
ALPHA LOCK	Places all alphabetic characters in uppercase until Alpha Lock is pressed again. Alpha Lock is a *toggle* key: it turns a function on and off.

Table 2.1 Function keys

Control Characters and Sequences

Pressed simultaneously with another key, the CTRL pad may alter the code of that key and initiate a function defined by your AOS/RT32 system. Throughout the manual, we denote a control character as CTRL-*x*, for example, CTRL-C. Unlike other characters, control characters and sequences can interrupt a program or the operation of your terminal. The CLI accepts and executes some control characters and sequences even if it is in the process of executing another CLI command.

The control characters recognized by the AOS/RT32 CLI are summarized in Table 2.2. None of the control characters enabled in AOS/VS by Screenedit Mode are included, since the AOS/RT32 CLI does not support Screenedit Mode.

You can affect the execution of a process that controls your terminal by typing a control character sequence. The first character of the sequence is always CTRL-C. The sequences of control characters recognized by the AOS/RT32 CLI are summarized in Table 2.3. AOS/VS users may notice that the control sequence CTRL-C CTRL-E, which terminates a process and creates a breakfile, is not included. AOS/RT32 does not support the creation of breakfiles in the CLI. In addition, the CTRL-C CTRL-C sequence does not apply to the CLI.

Control Character	Effect
CTRL-C	Begins a control character sequence. System action depends on the next character typed.
CTRL-D	Signifies end-of-file and causes an exception return to the task that performed the read operation. Two consecutive CTRL-Ds terminate the CLI process.
CTRL-O	Supresses display of data written to the terminal. Display resumes when (1) CTRL-O is depressed a second time; (2) a task performs a read or forced write to the console; or (3) the process terminates.
CTRL-P	Transmits the next character to a process without interpreting it. Two consecutive CTRL-Ps cause the system to interpret the first literally and pass the second to the task performing the read operation.
CTRL-Q	Resumes display of data at the terminal after it has been suspended with CTRL-S. Display resumes at the next page if a device is in Page Mode.
CTRL-S	Freezes terminal display. Display resumes when CTRL-Q is pressed or the current process terminates.
CTRL-T	Reserved.
CTRL-U	Cancels the current input line. Echos as ^ on hard-copy terminals.
CTRL-V	Reserved.

Table 2.2 Control characters

Control Sequence	Effect
CTRL-C CTRL-A	Interrupts a process that has defined a console-interrupt service task and transfers control to that task. If no such service task is defined, CTRL-C CTRL-A is ignored.
CTRL-C CTRL-B	Generates a console interrupt and aborts the process in control of the terminal.
CTRL-C CTRL-D through CTRL-C CTRL-Z	Reserved.

Table 2.3 Control character sequences

Command Line Syntax

Under AOS/RT32, command line syntax varies somewhat from that of the AOS/VS CLI. This section explains the uses of switches, arguments, delimiters, and terminators to construct valid CLI command lines, along with the coding aids accepted by the CLI.

The AOS/RT32 CLI prompts with two right parentheses,)). A command line is typed in immediately after the prompt in the form:

COMMAND */[switch] argument* ⤶

That is, a command line must consist of one of the commands listed in Chapter 5; may be followed by one or more command switches and arguments; and must include a valid terminator.

A long command line cannot be continued on the next line with the ampersand (&). Instead, the CLI accepts 256 characters before automatically terminating a statement. All characters in excess of this number are ignored, and a line-too-long message is transmitted to the display screen.

In interpreting commands, the CLI does not distinguish between uppercase and lowercase characters. Thus, the command line

)) TYPE FACTORIAL ⤶

is equivalent to

)) type factorial ⤶.

Command Switches

Commands can be modified with switches. A *switch* represents a processing option that modifies the execution of a command. Switches are optional. When used, they immediately follow the command in a command line. Arguments cannot be modified with switches in the CLI.

All switches begin with a slash (/) and take one of two forms: simple or keyword. A *simple switch* has the form:

/switch

and a keyword switch has the form:

/keyword=value.

The CHARACTERISTICS command, for example, can be modified with a simple switch—

))CHARACTERISTICS/PM ⤶

—which turns Page Mode on for your terminal, and with a keyword switch—

))CHARACTERISTICS/CPL=80 ⤶

—which sets the allowed number of characters per line to 80 for your terminal.

Arguments

An *argument* can be a filename, a process name, a device name, etc. Most AOS/RT32 CLI commands require arguments, some treat arguments as optional, and others do not allow them. Many commands allow multiple arguments in one command line. Nesting, however, is not supported in the CLI.

Delimiters

A command or command-switch combination must be separated from an argument with a *delimiter*. Arguments, too, must be delimited if more than one is used. The following characters or character combinations serve as delimiters in CLI command lines:

- one or more spaces,
- one or more tabs,
- a single comma, or
- any combination of spaces, tabs, and single commas.

So, for example, depending on the combination of delimiters chosen, a command line might look like one of the following.

```
)) TYPE EUCLID FACTORIAL ⤸
)) TYPE,EUCLID,FACTORIAL ⤸
)) TYPE   EUCLID   FACTORIAL ⤸
```

All of these forms are equivalent: the CLI interprets individual or groups of delimiters in a command line as a single comma.

Terminators

The CLI does not execute a command until it is terminated. The valid CLI terminators are:

New Line
Null
Form Feed (CTRL-L)
Carriage Return
End-of-file (CTRL-D)

On non-ANSI standard terminals, use Carriage Return for New Line.

You can specify a null argument by typing two consecutive commas or a delimiter followed by a terminator. (You might occasionally use null arguments in command lines when creating macros prior to system generation.)

Abbreviations

CLI commands and switches can be abbreviated. The shortest acceptable abbreviation is the smallest number of characters, beginning with the first character, that uniquely identifies the command or switch. For example, F is a valid abbreviation of the FILESTATUS command since no other commands begin with the same character. T, however, is not a valid abbreviation for the TIME command because another command, TYPE, also begins with T.

Although convenient, abbreviations are not the recommended form of issuing commands in macros that you intend to call from the CLI. Future revisions of AOS/RT32 may incorporate new CLI commands, rendering formerly unique abbreviations invalid.

Multiple-Command Input Lines

Two or more logically separate command lines can be entered on a single input line by separating each with a semicolon. The following example shows a dialogue with the AOS/RT32 CLI using multiple-command input lines.

```
))WHO 13;SUPERPROCESS ⤸
PID: 13    USER     SCAN_II
OFF

))SUPERPROCESS ON;BLOCK 13 ⤸
ON
#))
```

In this example, the user asks for the status of process 13 and the state of Superprocess Mode; the CLI returns the process information and reports that Superprocess Mode is off; and the user turns Superprocess on in order to block process 13. (Note that the CLI verifies the Superprocess setting and returns a prompt and pound sign, #)), before executing the BLOCK command.)

The CLI processes a multiple-command input line from left to right after the entire line is terminated. If you enter an invalid command in a multiple-command input line, or if the execution of a syntactically correct command causes an error or abort, the CLI will not execute commands that appear to the right of the offending command.

Error Message Handling

In AOS/VS, each error code is represented by a descriptive text string; the AOS/VS CLI uses this facility to display error messages for exceptional condition codes received during the execution of commands. To conserve memory space, the association of text string to error code is kept to a minimum under AOS/RT32. ERMES, the standard error message file provided with the AOS/VS system, is not maintained. Instead, the AOS/RT32 CLI returns twelve messages, covering the errors most commonly made by users in the CLI. These errors are primarily syntactical and are defined in Appendix A. The CLI responds to other errors with numeric code that the user must interpret by consulting Appendix B or the *AOS/RT32 System Programmer's Reference* (DGC No. 093-400016).

Differences Between the AOS/RT32 and AOS/VS CLIs

This chapter described several operational differences between the AOS/RT32 and AOS/VS CLIs—differences in control, command line syntax, and error message handling.

The AOS/RT32 CLI does not support the Break and Home function keys.

Control characters and sequences are unique to AOS/RT32 in these ways.

- Because Screenedit Mode is not supported, the CLI does not recognize CTRL-A, CTRL-B, CTRL-E, CTRL-F, CTRL-H, CTRL-I, CTRL-K, CTRL-X, or CTRL-Y.
- As a coding tool, CTRL-P is useful during system generation only.
- Because the CLI cannot create breakfiles, it does not support the control character sequence CTRL-C CTRL-E.
- The control character sequence CTRL-C CTRL-C is not supported by the CLI.

The AOS/RT32 CLI places the following conditions on command line syntax.

- Argument switches are not supported.
- Nesting is not supported. Thus, parentheses and angle brackets have no place in a command line.
- The continuation of a line with the ampersand character is not supported.
- The CLI accepts 256 characters before automatically terminating a statement. It ignores characters in excess of this number and transmits a line-too-long message to the console.

The CLI responds to the twelve most frequent operator errors with text strings, and otherwise returns numeric code.

Chapter 3

File and Macro Facilities

This chapter first discusses AOS/RT32 protocol for naming and referring to files, including generic filenames and device names. The features for monitoring, modifying, and displaying these files are covered next. The chapter then explains how macros must be constructed in order to execute under AOS/RT32. In conclusion, the differences between the file and macro facilities of the AOS/RT32 and AOS/VS CLIs are summarized.

Naming and Referring to Files

Each AOS/RT32 file has a name (called a filename and specified during system generation) that allows the user and the system to reference it. Once assigned, this name cannot be changed in the CLI. Names assigned to files are independent of the devices on which they may reside. An AOS/RT32 filename can consist of the following characters in any combination:

A through Z	uppercase alphabetic characters
a through z	lowercase alphabetic characters
0 through 9	numerics
$	dollar sign
_	underscore
?	question mark
.	period

The angle bracket, parentheses, and ampersand characters are reserved by the system and cannot be used as filename characters or coding aids. The CLI recognizes no difference between uppercase and lowercase alphabetic characters.

Two factors simplify file referencing in the AOS/RT32 CLI. First, AOS/RT32 does not support directories, pathnames, searchlists, or links; all files are referenced by their names. Second, the location of an AOS/RT32 file—in memory or on a device—is specified during the File Define phase of system generation. The CLI cannot delete a file or restrict it in any way.

With the exception of macronames, filenames always function as simple arguments in a command line. That is, accompanying directory files, templates, number signs, and backslashes are neither required nor supported.

Generic Filenames

Generic filenames refer to files or devices of a particular type, such as input and output files. The association between a generic filename and a specific file or device is made when you generate the CLI process. However, because AOS/RT32 does not allow processes to be initiated from the CLI, certain CLI parameters such as @LIST, @NULL, @DATA, and @CONSOLE are not supported.

Instead, input and output to the CLI is performed via the generic @INPUT and @OUTPUT files that you specify during system generation. Ordinarily, a single console serves as both the @INPUT and @OUTPUT file for an interactive process such as the CLI. In this case, the @INPUT echo function echos data to the @OUTPUT console. Alternatively, the system copies data from @INPUT to @OUTPUT when each represents a different file or device.

The generic filenames @INPUT and @OUTPUT acquire all the characteristics of the devices associated with them. Thus, if you associate the generic @OUTPUT file with a CRT terminal during system generation, the display of data at that terminal conforms to any specification that you set with the CHARACTERISTICS command. (See Chapter 5.)

Device Names

Like generic filenames, devices are defined when the system is generated. In the CLI, device names are most commonly referred to when data is transferred to or from a device, and the @ prefix always applies. AOS/RT32 supports record I/O to the devices in Table 3.1, and block I/O to the devices in Table 3.2.

Name	Device
@CON0	System Control Processor (SCP)
@CON2 through @CON*n*	DASHER® display terminals or asynchronous communications lines 2 through *n*

Table 3.1 Devices supported by AOS/RT32 for record I/O

Name[1]	Device
@DPD*(c)(u)* @DPG*(c)(u)*	Moving head disks *c* = 0 to 4 (up to 5 controllers) *u* = 0 to 7 (up to 8 units per controller)
@DPF*(c)(u)* @DPI*(c)(u)*	Moving head disks *c* = 0 to 7 (up to 8 controllers) *u* = 0 to 3 (up to 4 units per controller)
@LPB, @LPB1 to @LPB7	Data channel line printers 0 to 7
@MTA*(c)(u)* @MTB*(c)(u)*	Magnetic tape controllers[2] *c* = 0 or 1 (up to 2 controllers) *u* = 0 to 7 (up to 8 units per controller)
@MTC0 @MTC10	Magnetic tape contoller @MTC0 - first controller @MTC10 - second controller

Table 3.2 Devices supported by AOS/RT32 for block I/O

[1] *c represents the controller number; u represents the unit number.*

[2] *If c is not specified, its value is assumed to be 0.*

When referring to a magnetic tape unit from the CLI, you must specify the number of the tape unit and the file's position on the tape (its file number). The format is as follows:

@MTBx:y

where *x* is the number of the tape unit and *y* is the file number. Magnetic tape files are numbered sequentially, starting with 0. Thus, the device name @MTB0:2 specifies the third file on tape unit 0. If no file number is specified, the system refers to file 0, the first file on the tape, by default. (The format for such a reference is @MTB*x*.)

File-related Commands

This section describes the CLI commands for manipulating files and summarizes them in Table 3.3. As mentioned in Chapter 1, users can check a file's status, copy data to or from it, and display its contents with CLI commands. However, no file creation or deletion commands are supported.

The FILESTATUS command displays information about a file. Unlike its AOS/VS counterpart, this command requires at least one filename as its argument.

Used without switches, FILESTATUS confirms or denies the existence of a file. A number of switches provide additional information, among them,

- /ASSORTMENT, which furnishes both the size and type of a file, along with the date and time of its last modification. Currently, all AOS/RT32 files are user data files (UDF).
- /LENGTH, which returns a file's current size in bytes.
- /PACKET, which displays the entire contents of a file's ?FSTAT packet, allowing the user to examine all system information about the file.

These switches are particularly helpful before transferring data with the COPY command, described next.

Command	Argument(s)	Action
COPY	*dest-file sourcefile [sourcefile] ...*	Replaces data in one file or device with data from one or more others, or copies data to an empty device or file. The destination and source files must exist, and the destination file must be large enough to contain all data copied to it.
FILESTATUS	*filename [filename] ...*	Confirms or denies the existence of one or more files specifed in argument. Returns other file information requested in switches.
TYPE	*filename [filename] ...*	Displays the contents of one or more files.

Table 3.3 File-related commands

The COPY command replaces data in one file or device with data from another, or adds data to an empty device or file. The command requires at least two arguments: a destination file and one or more source files. These files must exist. If the CLI cannot locate the destination file, it does not create one but returns an error code number instead.

AOS/RT32 performs all block I/O on a block-by-block basis, beginning with the first byte. Thus, when the COPY command transfers data from a destination to a source file, it *replaces* the contents of the destination file with that data. If, for example, you copy a file named SOURCE with a current size of 10 bytes to a file named DEST with a current size of 20 bytes, the resulting size of DEST will be 10 bytes. In addition, if DEST is an uninitialized, memory-resident file, the system will distribute its unused memory resources elsewhere and return them to DEST when needed.

Because a file cannot expand beyond the size specified during system generation, the destination file must be as large or larger than the source file. Otherwise, the CLI transfers as much data to the destination file as its size permits, and then displays an error code number. To avoid this error, compare the current size of the source file with the maximum size of the destination file before copying. The FILESTATUS command, used with the /PACKET and /LENGTH or /ASSORTMENT switches described earlier, serves this purpose.

The COPY command can also append data to a file or device when the /A switch is supplied. In this case,

copying begins after the last byte of the destination file and increases its size by the number of bytes in the source file.

The TYPE command allows you to examine a file's contents or to verify that a COPY command has been executed. It accepts a filename as its argument and display's a file's contents at your console; it does not allow the contents to be altered.

The system displays file data at a much faster rate than the eye can read it. This rate of output can be controlled by altering the console's characteristics before issuing a TYPE command. The /ON and /PM switches, appended to the CHARACTERISTICS command, cause the system to suspend output after writing the allowed number of lines per page. To resume output, type CTRL-Q. (For details on this command and its switches, see Chapter 5.)

The Macro Facility

A macro, short for macroinstruction set, is a text file that consists of a sequence of commands. This section reviews the syntax and use of macros in the AOS/RT32 CLI.

An AOS/RT32 CLI macro is stored in a file and executed by typing the file's name. Like all other AOS/RT32 files, however, a macro file is created under AOS/VS and defined during system generation. The CLI executes macros but cannot create them. (You can, however, add to a macro with the COPY command and /A switch.) Valid syntax and legal code are especially important in AOS/RT32 macros because they can only be corrected under AOS/VS. Once the corrections are made, it is necessary to return to the File Define phase of system generation.

Macro Syntax

Macro filenames are specified during the File Define phase of system generation. They can, but need not, include a .CLI suffix. (Under AOS/VS, this suffix indicates that the file is a macro and tells the CLI to execute the commands within it.) A macro name should not duplicate any CLI command name since, if an ambiguity exists, the CLI will execute the command instead of the macro.

The following syntactical rules should be adhered to when constructing AOS/RT32 macros under AOS/VS.

- Command lines should not be continued with the ampersand character.
- Commands and arguments should not be nested.
- Only legal AOS/RT32 CLI commands and command switches should be used (see Chapter 5).
- Pseudo macros, range arguments, and dummy arguments should not be included because AOS/RT32 does not support them.
- Pathnames, linknames, and templates should not be supplied.

The following example illustrates the creation of a macro under AOS/VS using valid AOS/RT32 syntax.

```
) CREATE/I SAVE_IT.CLI ⤸
))TIME ⤸
))DATE ⤸
))WRITE COPYING INPUT DATA TO MAGTAPE 0:0 ⤸
))COPY @MTB0:0 SCANNER ⤸
))REWIND @MTB0 ⤸
))) ⤸
)
```

To execute this macro in the AOS/RT32 CLI, type SAVE_IT.CLI as specified during the File Define phase of system generation, followed by a New Line. (Alternatively, SAVE_IT.CLI could be defined SAVE_IT and referred to accordingly in the CLI.) This macro causes the system to

1. Output the date, time, and message to your terminal as follows;

   2:45:01
   26-JAN-82
   COPYING INPUT DATA TO MAGTAPE 0:0

2. Copy the contents of SCANNER to the first file on magnetic tape unit 0;
3. Rewind magnetic tape unit 0.

Thus, a macro can conveniently supplant any commonly executed series of CLI commands.

Differences Between the AOS/RT32 and AOS/VS CLIs

This section summarizes the many variations in AOS/RT32 CLI features discussed in this chapter. These variations occur in the CLI protocol for naming and referring to files, including generic filenames and device names; in file-related commands; and in the construction of macros.

The following conditions apply to the naming and referencing of files in AOS/RT32.

- Filenames are specified during the File Define phase of system generation only; the CLI cannot rename files.
- Directory files, pathnames, searchlists, and links are not supported.
- Access to files cannot be restricted in the CLI.
- The CLI does not support template characters, number signs, or backslashes.

The following conditions apply to generic filenames.

- Because AOS/RT32 does not allow the CLI to initiate other processes, the CLI parameters of @LIST, @NULL, @DATA, and @CONSOLE are not supported.
- The system copies CLI command lines from @INPUT to @OUTPUT. These generic files are set for the CLI during the Process Define phase of system generation.

Where device names are concerned, the following conditions apply.

- AOS/RT32 supports @CON0 and @CON2 through @CON*n* for record I/O only.
- AOS/RT32 supports moving head disks @DPD and @DPG (controllers 0 to 4 and units 0 to 7); moving head disks @DPF and @DPI (controllers 0 to 7 and units 0 to 3); data channel line printers @LPB and @LPB0 to @LPB7; magnetic tape controllers @MTA and @MTB (controllers 0 or 1 and units 0 to 7); and magnetic tape controller @MTC (where @MTC0 is the first controller and @MTC10 the second)--all for block I/O only.
- Fixed-head disks, card readers, and labeled magnetic tapes are not supported by AOS/RT32.

In the AOS/RT32 CLI, file-related commands differ from their AOS/VS conterparts in these ways.

- File access is not restricted in any way.
- The FILESTATUS command requires a filename as its argument and returns information about that file alone.
- Currently, the CLI reports all file types as UDF (user data file).
- The COPY command *replaces* data in a file or device with data from one or more others.
- The COPY command requires *existing* files as its arguments. Files, including destination files, cannot be created or deleted in the AOS/RT32 CLI.
- A file cannot expand beyond the size defined for it during system generation. Thus, when copying files, the destination file must be as large or larger than the source file.

AOS/RT32 places the following conditions on the construction of macros.

- Macro files must be created under AOS/VS and generated onto an AOS/RT32 system; the CLI executes but does not create them.
- The CLI does not recognize parentheses, angle brackets, or the continuation line symbol (&) in macro files.
- Pseudo macros, dummy arguments, and range arguments are not supported by AOS/RT32 and should not be used in macros intended for an AOS/RT32 system.

Chapter 4

Process and Device Control

This chapter explains the CLI features that control processes and devices. It begins by describing how processes are identified and referred to under AOS/RT32. The CLI commands for process control are presented next. The chapter then explains the use of CLI commands to control devices. It concludes with a summary of the differences between the AOS/RT32 and AOS/VS CLIs where process and device control are concerned.

Process Control

As explained in Chapter 1, processes are coded under AOS/VS and defined during the Process Define phase of an AOS/RT32 system generation. There are no parent or descendant, and no superior or inferior, processes in the AOS/RT32 environment. Instead, all processes reside in memory and compete for system resources based on a strict, priority-driven schedule. A process may be blocked or unblocked when you bring up an AOS/RT32 system, but all processes come up concurrently.

In an environment such as this, where events are predetermined, complex, and often parallel, the CLI functions mainly as a monitor and scheduler of processes. This section first explains how processes are referenced in the CLI, based on the way that the system and system programmer identify them. Then the CLI commands for process monitoring, scheduling, termination, and communication are described.

Identifying and Referring to Processes

During the system generation procedure, a *process name* is assigned to each process. This name consists of up to 15 valid filename characters, which include the letters A through Z; the numbers 0 through 9; the period (.), dollar sign ($), and question mark (?); and the underscore (_). Uppercase and lowercase characters are treated alike.

A process is also identified by a *process identification number*, or PID. The AOS/RT32 system assigns each process a unique PID number in the range from one through 255. (The system process is PID 0.) PID numbers are assigned sequentially. If five processes reside in your AOS/RT32 environment, the system numbers them one through five. If one of these processes is terminated, the remaining four retain their original PIDs; the system does not renumber them.

Finally, the system assigns each process a *username*. This name appears simply as "username", and is the same for all processes.

In the CLI, a process is generally referred to by its name or its PID. Either can serve as an optional or required argument in a process-related command line. The process name and PID are always specified exactly as they were defined; the components of a process's identity are constant throughout the life of that process.

Process Monitoring

The CLI's process monitoring commands supply information about a process's identity. These commands are presented in Table 4.1 and descibed below.

Command	Argument(s)	Action
WHO	*[procname / pid]* ...	Displays PID, username, and process name of one or more processes. Program names are not reported.
?	None	Displays PIDs, usernames, and process names of all processes. Program names are not reported.

Process monitoring commands

The WHO and ? commands are especially useful for identifying the sender of a message to the CLI process; for confirming or denying the existence of a process; and for verifying the identity of a process before blocking, unblocking, or terminating it.

The WHO command returns the PID, username, and process name of one or more processes; no program names are displayed. The command accepts, but does not require, one or more arguments. When no argument is specified, WHO applies to the CLI. The argument can refer to a blocked, but not a terminated, process. The information displayed is the same regardless of whether the process is blocked or unblocked. If a terminated or nonexistent

process is used as an argument, the system returns an error message to your console: ATTEMPT TO ACCESS NONEXISTENT PROCESS.

The ? command is like a WHO command multiplied by the number of processes in your system: it displays the PID, username, and process name of every process in the AOS/RT32 environment. It does not indicate whether a process is blocked or unblocked. Terminated processes are not reflected in the listing.

Process Scheduling

The CLI supports several process scheduling features as a means of responding to external events. The commands for scheduling processes are summarized in Table 4.2 and described below.

Command	Argument(s)	Action
SUPERPROCESS	*[on / off]*	Controls the setting of Superprocess Mode, or displays the setting when no argument is supplied. When on, causes output of pound sign with prompt, #)), and allows process blocking, unblocking, and termination.
BLOCK	*procname / pid*	Causes suspension of all tasks within one or more processes.
UNBLOCK	*procname / pid*	Frees all tasks within one or more processes from their suspended state. Applies only to previously blocked processes.
PAUSE	*seconds / seconds.milliseconds*	Delays the CLI for the amount of time specified in the argument, where seconds is a number between 0 and 65335, and milliseconds is a number between 0 and 999.

Table 4.2 Process scheduling commands

The SUPERPROCESS command monitors and controls the setting of Superprocess Mode. Unless this setting is on, the CLI cannot block, unblock, or terminate another process. However, since all users are superusers under AOS/RT32, no special privilege is required to issue this command. The command allows, but does not require, one argument; when none is supplied, the CLI reports the Superprocess setting. When this setting is on, the CLI outputs the prompt, #)).

When used only as needed, the SUPERPROCESS command serves as a safety feature: it delimits the blocking, unblocking, and termination procedures from other CLI operations, reminding users to execute them with care.

Although many processes include code to block and unblock themselves, external circumstances often occasion the need to control them from the CLI. The BLOCK, UNBLOCK, and PAUSE commands give CLI users the latitude needed to (1) respond to variations in the external environment; (2) synchronize internal with external events; and (3) free up the system's resources for another process's use.

The BLOCK command suspends all tasks within a process in order to block its execution. The command requires one or more process names or identifiers as its argument. Unlike its AOS/VS counterpart, which executes for superior or privileged processes only, the AOS/RT32 version of BLOCK can be issued by any user to suspend any process.

> **CAUTION:** *Once the CLI is blocked, another process must unblock it.*

The UNBLOCK command releases all tasks within a process from their suspended state. Thus, UNBLOCK applies only to a process that has been suspended by the system, by another process, or by its own request. At least one process name or identifier must be supplied in argument.

The PAUSE command is useful for synchronizing the execution of processes with each other and/or with external events. It is also useful in macros that affect scheduling, where the variables involved are stable and known.

The action of the PAUSE command can be likened to process blocking and unblocking, with the distinction that PAUSE is a self-contained function that affects only the CLI. PAUSE delays the CLI for the amount of time specified in its required argument.

PAUSE is most often issued in conjunction with at least one other command. For this reason, and because PAUSE delays the CLI, it is generally issued on the same line as the command scheduled to follow it; that is, both are typed in multiple-command input-line form. The following is an example.

```
)) PAUSE 300;SUPERPROCESS ON;BLOCK PROG3 ↓
```

Process Termination

The CLI's process termination commands allow you to conclude a planned sequence of events, or to cancel those events if the need arises. The commands for terminating processses are shown in Table 4.3 and described below.

Command	Argument(s)	Action
BYE	None.	Terminates the CLI. No termination message is displayed. To restart the CLI, return to the Process Define phase of the system generation procedure.
TERMINATE	*procname / pid*	Kills all tasks within one or more processes, releasing their resources for use by existing processes.

Table 4.3 Process termination commands

The TERMINATE command kills all tasks within a process, purging the system of the process. The resources of a terminated process are released for use by other, existing processes. The command requires at least one process name or identifier as its argument.

The BYE command terminates the AOS/RT32 CLI. No argument is allowed, and no termination message is displayed. Either the BYE or TERMINATE commands can be used to terminate the CLI; there is no difference in the way resources are released to the system. Once terminated, however, the CLI can only be restarted by returning to the Process Define phase of system generation.

Process Communication

Process communication enables users to coordinate their scheduling efforts. The commands for process communication are shown in Table 4.4. SEND is discussed below; WHO was discussed earlier in this section.

Command	Argument(s)	Effect
SEND	*procname / pid / console message*	Sends a message to the process specified in the first argument.
WHO	*[procname / pid] ...*	Identifies the sender of a message or the process you would like to send a message to.

Table 4.4 Process communication commands

The SEND command allows operators to pass important textual information to and from process consoles. It can also be used in macros that implement routine scheduling procedures.

The SEND command requires two arguments—the first a process name, console name, or process identification number, and the second a message of no more than 256 characters. This message should not include commas, tabs, or control characters. The system converts commas and tabs to spaces when it sends your message, and it cannot send control characters. Because AOS/RT32 does not support the nesting of commands or arguments, SEND must be issued in multiple-command input line form when sending a message to more than one user at a time.

A PID number identifies the sender of a message to the process that receives it. Further identification is obtained with the WHO command.

Device Control

Like processes, devices are defined during the system generation procedure. Chapter 3 explains how devices are referenced from the CLI, and Tables 3.1 and 3.2 list the devices that AOS/RT32 supports. Chapter 3 also describes the COPY command, which transfers data from one file or device to another, from a file to a device, and from a device to a file. This section describes the CLI commands provided for the control of devices as physical entities. These commands are summarized in Table 4.5.

Command	Argument(s)	Effect
CHARACTERISTICS	*[device name] ...*	Sets the characteristics of a device according to information supplied in the switches. When no switches are supplied, the features of the given or default device are displayed. When no device name is given, the command applies to your console.
COPY	*dest-file sourcefile [sourcefile] ...*	Replaces data in one file or device with data from one or more others, or copies data to an empty device or file. The destination and source files must exist, and the destination file must be large enough to contain all data copied to it.
REWIND	*tapeunit [tapeunit] ...*	Rewinds one or more tapes.

Table 4.5 Commands that control devices

Character devices perform I/O in byte units. CRTs, hard-copy terminals, and line printers are typical character devices.

As mentioned in Chapter 2, character devices can operate in either Binary or Text Mode. (Text Mode is the default mode.) When a character device is in Text Mode, the system interprets each byte according to the device's characteristics or distinguishing features. You can set or display these features with the CHARACTERISTICS command.

The CHARACTERISTICS command is especially useful for formatting the output of a device and for controlling the amount of information it displays. The command allows, but does not require, one or more device names as its argument. When no argument is supplied, the command applies to your console. When no command switches are supplied, the system simply displays the characteristics of the given or default device. The characteristics that you set remain in effect until you change them or terminate the CLI.

With four exceptions (described in Chapter 5), individual characteristics are activated with the /ON switch and deactivated with the /OFF switch. /ON and /OFF affect the switches that *follow* them, not the switches that precede them. Since /ON is set automatically, it need not be included in a command line as long as you specify the characteristics you want to activate first. However, once the system encounters an /OFF switch or delimiter, /ON is no longer automatically assumed and must be supplied to activate subsequent characteristics.

Your system may have defined one or more magnetic tape units for data storage. The COPY command, shown in Table 4.5, can transfer data to a tape unit from memory or from another device. This transfer is often followed by a rewind procedure. The CLI provides the REWIND command for this purpose.

The REWIND command provides a convenient way to reposition a magnetic tape to the beginning of a tape. It is often used in macros that implement routine data transfer and storage procedures. The command requires one or more arguments that specify the tape unit(s) you want to rewind. (The protocol for referring to tape units and the files stored on them is explained in Chapter 3.)

Differences Between the AOS/RT32 and AOS/VS CLIs

This section summarizes the differences between the AOS/RT32 and AOS/VS CLIs where process and device control are concerned.

The AOS/RT32 protocol for creating, naming, and referring to processes is as follows.

- The CLI cannot create processes.
- Once defined by the system and system programmer, process names, identification numbers, and usernames never change. The system assigns the same username ("user") to all processes.

The AOS/RT32 CLI commands for process monitoring differ from their AOS/VS counterparts in these ways.

- Features for setting or displaying a process's type and priority, and for displaying a system's hostname or a process's runtime information, are not supported.
- The WHO and ? commands report process identification numbers, usernames, and process names only. Program names are not displayed.

The process scheduling commands of the AOS/RT32 CLI are unique in the following ways.

- The SUPERPROCESS command is a function of safety rather than privilege: any CLI user can execute this command.
- The BLOCK and UNBLOCK commands execute for any user when Superprocess Mode is on.

The following conditions apply to process termination in the AOS/RT32 CLI.

- The TERMINATE command executes for any user when Superprocess Mode is on.
- Once terminated with the BYE or TERMINATE commands, the CLI cannot be restarted.
- The BYE command does not return a termination message.

The following process communication features are unique to the AOS/RT32 CLI.

- Since nesting is not supported, it is necessary to use multiple-command input line form when issuing messages to more than one process at a time with the SEND command.
- Since the ampersand is interpreted literally, it should not be used to continue a line in a message issued with the SEND command.
- The sending of IPC (Interprocess Communication) messages with the CONTROL command is not supported.

The following conditions affect the control of devices in the AOS/RT32 CLI.

- The CLI cannot assign or deassign a character device for a process's exclusive use with ASSIGN and DEASSIGN commands.
- The CLI cannot send mount or dismount messages to an operator, or perform load and dump operations, except via the COPY command.

Chapter 5

CLI Command Dictionary

This chapter summarizes all CLI commands in table form, and then describes each command in alphabetical order. The conventions used in the sample formats and command lines are described in the Preface. Readers may find it helpful to review these conventions before proceeding.

Table 5.1 summarizes the CLI commands supported by AOS/RT32. The table also shows sample formats for the commands. These formats do not include optional switches. For details on any command and the switches that may apply to it, see its description later in the chapter.

Command and Sample Format	Action
?	Displays information about all processes
BLOCK *procname / pid*	Blocks a process
BYE	Terminates the CLI
CHARACTERISTICS *[device-name]*	Sets or displays device characteristics
COPY *dest-file sourcefile [sourcefile] ...*	Copies one or more files to a destination file
DATE *[month day year]*	Sets or displays the system date
FILESTATUS *filename [filename] ...*	Reports status of one or more files
HELP	Lists all CLI commands
PAUSE *seconds / seconds.milliseconds*	Delays the CLI
REWIND *tapeunit [tapeunit] ...*	Rewinds one or more tapes
SEND *procname / pid / consolename message*	Sends a message to another terminal
SUPERPROCESS *[on / off]*	Sets or displays the Superprocess setting
TERMINATE *procname / pid*	Terminates a process
TIME *[hours minutes seconds]*	Sets or displays the system time
TYPE *filename [filename] ...*	Displays the contents of one or more files
UNBLOCK *procname / pid*	Unblocks a previously blocked process
WHO *[procname / pid]...*	Displays information about a process
WRITE *[string]*	Displays string

AOS/RT32 CLI commands

?

Display information on all processes

?

This command allows no arguments. It displays the process identification numbers (PIDs), usernames, and process names of all processes defined in the AOS/RT32 environment. Program names are not included in the listing.

Examples

))? ⤸

PID 1	*USER*	*SCANNER*
PID 2	*USER*	*COMPOSITE*
PID 3	*USER*	*NUMBER_CRUNCH*
PID 4	*USER*	*MARINO*
PID 5	*USER*	*BORNS*

Table 5.1

Display the process identification numbers, usernames, and process names of all processes.

BLOCK

Block a process

BLOCK *procname* / *pid*

This command requires a process name or process identification number (PID) as its argument. The system will not block a process unless Superprocess Mode is on.

Examples

#))BLOCK 19 ⤸

Block the process whose process identification number is 19. The pound sign (#) indicates that Superprocess Mode is on.

#))BLOCK PROG3 ⤸

Block the process named PROG3. Again, note the pound sign (#), indicating that Superprocess Mode is on.

BYE

Terminate this CLI process

BYE

This command terminates the CLI and returns its memory resources to the AOS/RT32 system. No termination message is displayed. Once terminated, the CLI cannot be restarted.

Examples

))BYE ⤸

Terminate the CLI.

CHARACTERISTICS

Set or display device characteristics

CHARACTERISTICS*[/switch] [device-name] ...*

Device characteristics control the way a character device interprets input and output. The characteristics you set remain in effect until you change them or log off the system. CHARACTERISTICS commands can be issued in succession. When no argument is supplied, your terminal becomes the default device. When no switches are supplied, the system simply displays the characteristics of the specified or default device. Tables 5.2 and 5.3 list the command switches that apply to CHARACTERISTICS.

Examples

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

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

))CHARACTERISTICS/LPP=24 ⤸

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

))CHARACTERISTICS/OFF/NRM/ON/PM/ULC
CON12 ⤸

Allow the device named CON12 to receive SEND messages by deactivating /NRM; turn Page Mode on; and accept both uppercase and lowercase on input.

Command Switch	Effect
/8BT	When on, causes all 8 bits of an ASCII character to be interpreted as data. Regardless of whether this switch is set or not, the following octal codes will echo an up-arrow followed by an alphabetic character: 1 to 10; 13; 16 to 32; and 34 to 37.
/CPL=*n*	Sets the characters per line, in decimal, to the number specified by *n*.
/DEFAULT	Used alone, this switch displays the default characteristics of a device. Used with other switches, it sets the default characteristics of a device.
/EB0	For echoing to occur on your terminal, you must set /EB0 or /EB1. When on, /EB0 echoes control characters such as ↑A and ↑B. It echoes ESC as $. For more information, see the description of the ?GCHR system call in the *AOS/RT32 System Programmer's Reference.*
/EB1	When on, echoes characters exactly as they are input. For more information, see the description of the ?GCHR system call in the *AOS/RT32 System Programmer's Reference.*
/EOL	When this switch is on, the device will not output a new line if the allowed number of characters per line (CPL) is exceeded on input.
/ESC	When on, this switch causes the ESC character to produce a console interrupt (^C^A), as long as the ?INTWT call has defined a console interrupt task. (See the *AOS/RT32 System Programmer's Reference* for details.)
/FF	When on, outputs a form feed on open.
/FKT	When on, permits any function key to serve as a delimiter in data-sensitive read operations. **Warning:** *This switch does not apply to CLI commands. Use only valid terminators to end them.*
/LPP=*n*	Sets the lines per page, in decimal, to the number specified by *n*.
/LT	If this switch is on, the system outputs 60 (decimal) nulls on open and close.
/MOD	When on, specifies that a device is on a modem interface. This switch is set during system generation and cannot be changed thereafter.
/MRI	When on, enables the Monitor Ring Indicator. This switch is set during system generation and cannot be changed thereafter.
/NAS	If this switch is on, the device is considered non-ANSI standard. On input, /NAS converts carriage return to new line and line feed to carriage return. On output, it converts line feed to carriage return-line feed.
/NNL	When on, this switch tells the system not to automatically append new lines to card images.
/NRM	When on, prevents a terminal from receiving SEND messages.
/OFF	Followed by one or more switches, the /OFF switch sets each succeeding characteristic to OFF until the system encounters an /ON switch or a delimiter. Unlike the /ON switch, the /OFF switch is not set automatically.
/ON	Followed by one or more switches, the /ON switch sets each succeeding characteristic to ON until the system encounters an /OFF switch or delimiter. The /ON switch is automatically set unless it follows an /OFF switch, and is optional for this reason.
/OTT	When this switch is on, it converts octal 175 and 176 to octal 33 on input.
/PBN	When on, enables packed format on binary read; 4 columns are put in 3 words. If this switch is off, columns are right-justified in memory (card readers only).
/PM	This switch controls Page Mode. If it is on, output is suspended when the allowed number of lines per page (LPP) is reached. Output resumes when the user types CTRL-Q.
/RAC	If this switch is on, two rubouts are sent after each new line and carriage return.
/RAF	If this switch is on, 21 (decimal) rubouts are sent after each form feed.
/RAT	If this switch is on, two rubouts are sent after each tab.
/RESET	Sets the characteristics of a device to its default characteristics. This switch must be used alone.
/SFF	When on, this switch causes simulation of form feed.
/ST	When on, causes simulation of a tab stop at every eighth column.
/TO	Enables time-outs when on.
/TSP	Trailing spaces are included when this switch is on and suppressed when it is off (card readers only).
/UCO	If this switch is on, lowercase is converted to uppercase on output.
/ULC	If this switch is on, both uppercase and lowercase are accepted on input. If this switch is off, lowercase input is converted to uppercase.
/WRP	When on, this switch causes hardware to generate a new line when the allowed number of characters per line (CPL) has been reached.

Table 5.2 CHARACTERISTICS switches

CHARACTERISTICS switches (continued)

You can identify your terminal using CHARACTERISTICS with any of the switches listed in Table 5.3.

Command Switch	Identifies
/HARDCOPY	Hard-copy terminals
/4010I	DGC Model 4010I terminals
/6012	DGC Model 6012 terminals
/605x	DGC Model 6052 or 6053 terminals
/CRT3	All other video display terminals

Table 5.3 Terminal indentification switches

COPY

Copy one or more files to a destination file

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

This command requires two *existing* files as its arguments. If the destination file supplied in the command line does not exist, the CLI does not create one but returns an error code number instead. Because the destination file cannot expand beyond the maximum size defined for it during system generation, it must be large enough to contain all data copied to it. Otherwise, the CLI copies as much data to the destination file as its size permits, and then returns an error code number.

All block I/O is performed on a block-by-block basis, beginning with the first byte. The result is a *replacement* of data in the destination file with data from one or more sources. When the /A switch is used, copying begins after the last byte of the destination file. In this case, the system adds to, rather than replaces, the contents of the destination file. Table 5.4 presents the switches, including the /A switch, that modify the COPY command.

Command Switch	Effect
/A	Appends new data to the existing data in the destination file.
/IDENSITY=*mode*	Controls the magnetic tape density for input files. Use this switch with MTB, model 6026 tape drives only. Mode options include Mode Density 800 800 BPI 1600 1600 BPI ADM Automatic Density Matching
/IMTRSIZE=*blocksize*	Controls the magnetic tape block size for input files. Minimum block size is 4 bytes.
/ODENSITY=*mode*	Controls the magnetic tape density of output files. Use this switch with MTB, model 6026 tape drives only. Mode options include Mode Density 800 800 BPI 1600 1600 BPI ADM Automatic Density Matching
/OMTRSIZE=*blocksize*	Controls the magnetic tape block size for output files. Minimum block size is 4 bytes.

Table 5.4 COPY switches

Examples

))COPY OUTPUTFILE TESTRUN ↓

Copy the source file TESTRUN to the destination file OUTPUTFILE, where both are existing files and OUTPUTFILE is as large or larger than TESTRUN. Copying begins at the first byte of OUTPUTFILE.

))COPY/A MASTERFILE TEST1 TEST2 TEST 3 ↓

Append TEST1, TEST2, and TEST3 to MASTERFILE, where all are existing files and MASTERFILE is large enough to contain the other three. In this example, copying begins after the last byte of MASTERFILE.

))COPY SIMULATOR @MTA0:0 ↓

Copy the first file on magnetic tape unit 0 to SIMULATOR, where both are previously defined files and SIMULATOR is the larger file.

DATE

Set or display the current system date

DATE *[month day year]*

When no argument is supplied, the system date is displayed. Entering the month, day, and year sets the date. Each argument consists of one or two digits.

Examples

))DATE 11 26 82 ⤸

Set the system date to November 26, 1982 using two-digit arguments.

)) DATE 4 1 82 ⤸

Set the system date to April 1, 1982 using one- and two-digit arguments.

))DATE ⤸
04-APR-82

Display the current system date.

FILESTATUS

Display status information for one or more files

FILESTATUS*[/switch] filename [filename] ...*

This command requires at least one filename as its argument. When used without switches, FILESTATUS simply confirms a file's existence by returning its name, or denies the file's existence by returning a prompt.

Use the command switches listed in Table 5.5 to obtain other information about specific files.

Command Switch	Effect
/ASSORTMENT	Displays an assortment of file information, including file type, date and time of creation, and file length.
/DCR	Displays file creation date.
/DLA	Displays date file was last accessed.
/DLM	Displays date file was last modified.
/ELEMENTSIZE	Displays the number of disk blocks in a file element for this file. A file element is the smallest unit by which a disk file can grow. (A disk block is 512 bytes.)
/LENGTH	Displays a file's length in bytes.
/PACKET	Displays the entire contents of the packet returned by the ?FSTAT system call. This packet contains such information as the file element size, date and time of creation, date and time of last modification and access, and file size in bytes. (For more details, see the description of ?FSTAT in the *AOS/RT32 System Programmer's Reference.*)
/TCR	Displays the time of a file's creation.
/TLA	Displays the time a file was last accessed.
/TLM	Displays the time a file was last modified.
/TYPE	Displays the file's type as a three-character mnemonic. (Currently, the CLI reports a file type of UDF, user data file, for all files.)

Table 5.5 FILESTATUS switches

Examples

))FILESTATUS FACTORAIL ⤸
))

Confirm or deny the existence of a file named FACTORAIL. In this case, existence is denied, as indicated by the prompt)).

))FILESTATUS FACTORIAL ⤸
FACTORIAL

Confirm or deny the existence of the file named FACTORIAL. In this case, existence is confirmed.

))FILESTATUS/ASSORTMENT EUCLID ⤸

EUCLID UDF 3-JAN-82 11:00:18 27014

Display the type, date and time of creation, and byte size of the file named EUCLID.

HELP
List all CLI commands

HELP

Unlike its AOS/VS counterpart, this command accepts no arguments or switches and displays a simple list of the CLI commands supported by AOS/RT32.

Examples

))HELP ⤸

?
BLOCK
BYE
CHARACTERISTICS
COPY
DATE
FILESTATUS
HELP
PAUSE
REWIND
SEND
SUPERPROCESS
TERMINATE
TIME
TYPE
UNBLOCK
WHO
WRITE

Display all CLI command names.

PAUSE
Delay the CLI

PAUSE *seconds* / *seconds.milliseconds*

The argument required by this command can be formatted in seconds (where seconds is a number between 0 and 65535) or in seconds.milliseconds (where milliseconds is a number between 0 and 999).

Examples

))PAUSE 5 ⤸

Delay the CLI for five seconds.

))PAUSE 8.50 ⤸

Delay the CLI for 8.5 seconds.

REWIND

Rewind one or more tapes

REWIND *tapeunit [tapeunit] ...*

Unlike its AOS/VS counterpart, this command requires only a tape unit's name and number as its argument. No linkname should be substituted as an argument in the command line.

Examples

```
))REWIND @MTA0 ⤸
```

Rewind the magnetic tape on unit @MTA0.

SEND

Send a message to a terminal

SEND *procname | pid | consolename message*

This command enables you to send a message to another process where the target process is any with a terminal. The command takes either a process name, a process identification number (PID), or a console name as one argument, and a message as the other. The message should not include commas, tabs, or control characters. The system converts commas and tabs to spaces when sending a message, and it cannot send control characters.

Examples

```
))SEND 2 OKAY TO UNBLOCK SCANNER? ⤸
```

Send a message to the process with a PID of 2.

```
FROM PID 8: OKAY TO UNBLOCK SCANNER?
))SEND 8 GO AHEAD. ⤸
```

Send a response to the message received from the process whose PID is 8.

```
))SEND CON5 SYSTEM GOING DOWN IN 10
MINUTES;SEND HARRINGTON SYSTEM GOING
DOWN IN 10 MINUTES ⤸
```

Send messages to the console named CON5 and the process named HARRINGTON.

SUPERPROCESS

Set or display the Superprocess setting

SUPERPROCESS *[on | off]*

This command controls the CLI's Superprocess setting, which allows or prevents process manipulation depending on whether it is on or off. Thus, the Superprocess feature acts as a kind of safety catch, preventing the accidental blocking, unblocking, or termination of a process. The command accepts an argument of *on* or *off*, and sets the Superprocess Mode accordingly. The CLI precedes each prompt with a pound sign (#) when Superprocess is turned on. When no argument is supplied, the CLI simply reports the current Superprocess setting.

Examples

```
))SUPERPROCESS ↓
OFF
))SUPERPROCESS ON ↓
ON
#))
```

First, display the current Superprocess setting. Second, turn Superprocess on.

> **NOTE:** *When Superprocess is on, the CLI outputs the prompt #)) in the left margin. When you supply an argument in the command line, the Superprocess setting is automatically verified.*

TERMINATE

Terminate a process

TERMINATE *procname | pid*

This command requires a process name or a process identification number (PID) as its argument. No process can be terminated, however, unless Superprocess Mode is on.

Examples

```
#))TERMINATE 17 ↓
```

Terminate the process whose PID is 17. The pound sign (#) preceding the prompt indicates that Superprocess Mode is on. The CLI does not execute the TERMINATE command unless Superprocess Mode is on.

```
#))TERMINATE SIMULATOR ↓
```

Terminate the process named SIMULATOR. Again, the pound sign indicates that Superprocess Mode is on.

TIME

Set or display the current system time

TIME *[hours minutes seconds]*

This command displays the current system time when no argument is supplied. To set the time, specify hours, minutes, and seconds, where each argument is a two-digit number. The minute and second arguments are optional.

Examples

))TIME ↵
2:45:00

Display the current system time.

))TIME 3 05 ↵

Set the system time to 3:05 PM.

TYPE

Display a file's contents

TYPE *filename [filename] ...*

This command causes the display of a file's contents at your console; it does not allow the contents to be altered.

Examples

))TYPE MACROFILE ↵
TIME
DATE
WRITE COPYING INPUT DATA TO MAGTAPE 0:0
COPY @MTB0:0 SCANNER
REWIND @MTB0

Display the contents of MACROFILE.

UNBLOCK

Unblock a previously blocked process

UNBLOCK *procname | pid*

This command requires at least one process name or process identification number (PID) as its argument. The process to be unblocked must have been blocked previously, and Superprocess Mode must be on.

Examples

```
))SUPERPROCESS ON ⤸
ON
#))BLOCK 17;PAUSE 300;UNBLOCK 17 ⤸
```

First, turn Superprocess Mode on. (The CLI verifies the Superprocess setting and returns a prompt preceded by the pound sign.) Then block process 17; delay the CLI for 300 seconds (five minutes); and unblock process 17.

WHO

Display process information

WHO *[procname | PID] ...*

This command accepts a process name or a process identification number (PID) as its argument, and returns the PID, username, and process name of that process. No program names are displayed. When no argument is specified, the command applies to the CLI.

Examples

```
))WHO ⤸
PID: 10    USER    FITZGERALD
```

Display the user's PID (10), username (USER), and process name (FITZGERALD).

```
))WHO 3 ⤸
PID: 3    USER    MARINO
```

Display the PID, username, and process name for the process with a PID of 3.

```
))WHO ACES ⤸
PID: 7    USER    ACES
```

Display the PID, username, and process name for the process named ACES.

WRITE

Display string

WRITE *[string]*

The WRITE command accepts, but does not require, a string argument and returns that string to your console. It is most often used in macros as a form of commentary on the macroinstructions being executed. When no argument is supplied, an empty string is returned to your terminal.

Examples

```
))TIME;WRITE;DATE ↓
2:45:00

26-JAN-82
```

Display the system time, write an empty string (skip down a line), and display the system date.

```
))WRITE COPYING TESTRUN TO MASTERFILE ↓
COPYING TESTRUN TO MASTERFILE
```

Display a string of text.

Appendix A

Exceptional Condition Messages

This appendix describes the exceptional condition messages that you might receive after entering an invalid AOS/RT32 CLI command line. These messages, or text strings, are associated with error code numbers that pass from the system to AC0. The messages cover the errors most commonly made by CLI users. For all other errors, the system returns code numbers that you can interpret by consulting Appendix B.

If you are testing a program under AOS/VS, see the *AOS/VS Programmer's Manual* (DGC No. 093-000241) for details on runtime errors. If you receive an error while running a program under AOS/RT32, consult the *AOS/RT32 System Programmer's Reference* (DGC No. 093-400016). For an explanation of errors that occur while you are running a utility under AOS/VS, see the manual for that utility.

Messages

ATTEMPT TO ACCESS NONEXISTENT PROCESS

The system responds with this message when you attempt to access a terminated or undefined process, or when you mistype the process name or PID in a command line. Use the ? or WHO commands to verify a process's identity.

CALLER NOT PRIVILEGED FOR THIS ACTION

This message occurs when you attempt to block, unblock, or terminate a process without first turning Superprocess Mode on.

NOT A COMMAND OR MACRO

This message is returned when you mistype a command or macro name, when you specify an undefined macro file, or when you attempt to issue a CLI command not supported by AOS/RT32.

COMMAND REQUIRES ARGUMENTS

This message indicates that one or more required arguments are missing from your command line.

COMMAND ABBREVIATION NOT UNIQUE

The system reports this message when it requires an additional character or two to identify the command you are referring to.

ARGUMENT MISSING

This message is returned in cases where a command requires a single argument that you have failed to specify.

WRONG NUMBER OF ARGUMENTS

This message occurs when you have supplied only one argument with a command that requires two or more, or when you have supplied but have not correctly delimited those arguments.

ILLEGAL DECIMAL NUMBER

With this message, the system indicates that (1) the argument you supply must be an unsigned, positive decimal number; or (2) your argument must conform to a recognizable format.

NULL SWITCH ENCOUNTERED

The system responds with this message when it encounters a slash not followed by a switch, a keyword, or a value.

UNKNOWN SWITCH

A switch is "unknown" when it has been mistyped, when it has been mistakenly delimited from the slash that precedes it, or when the AOS/RT32 CLI does not support it.

SWITCH NOT UNIQUE

The system reports this message when it requires an additional character or two to identify the switch you are referring to.

DUPLICATE SWITCHES

This message indicates that you have supplied two or more of the same switch in a command line.

Appendix B

Exceptional Condition Codes

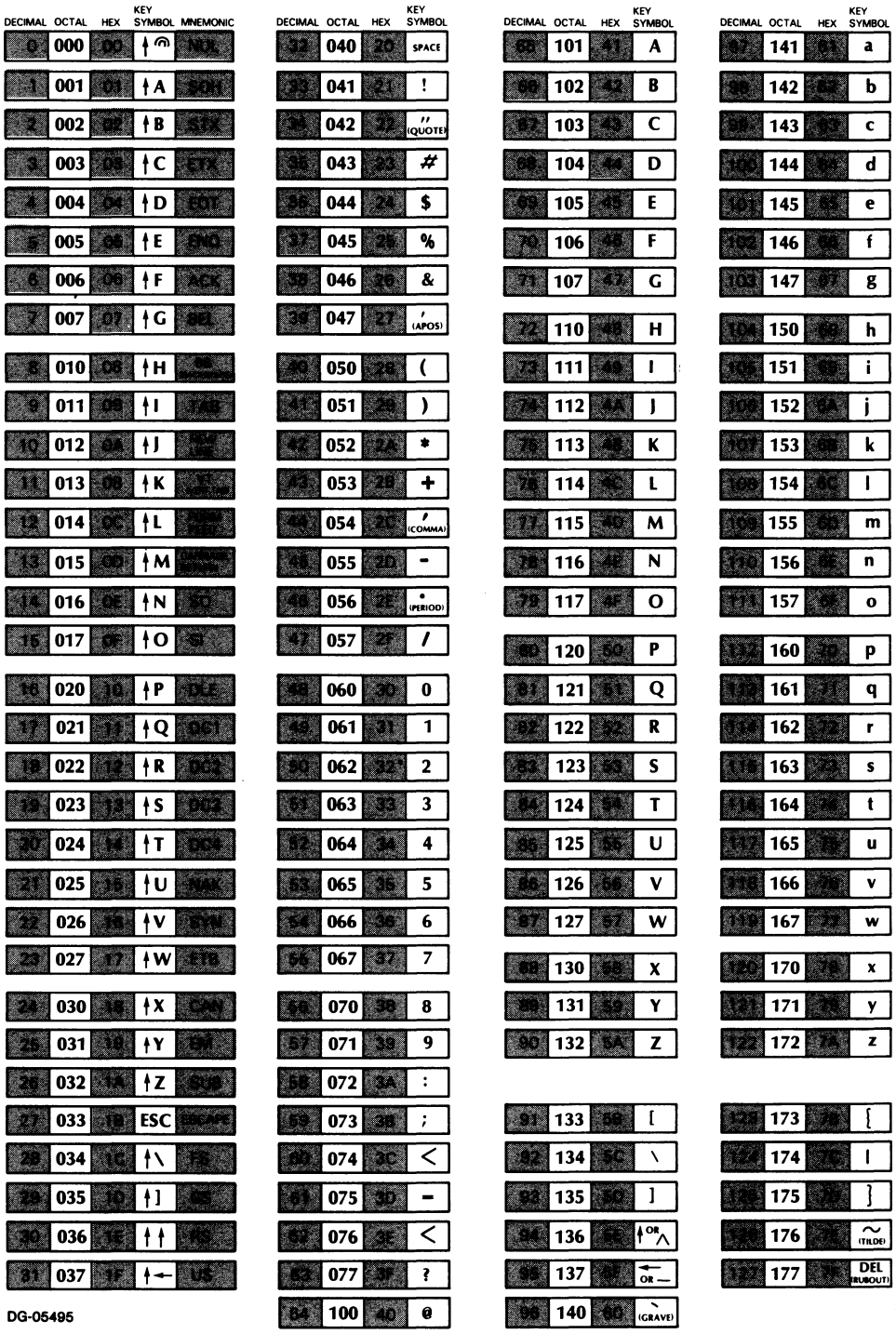
When a CLI command fails, AOS/RT32 passes an error code in AC0 and returns an error code number to your console. This number indicates the reason for the error and is accompanied by a textual message, referring the user to the *AOS/RT32 System Programmer's Reference* for details. For the convenience of CLI users, Table B.1 lists the most commonly received error codes according to their numeric assembled values.

Octal Value	Code	Message
1	ERICM	Illegal system command
2	ERFNO	Channel not open
5	ERMEM	Insufficient memory available
10	ERTIM	Illegal time argument
11	ERNOT	No task control block available
12	ERXMT	Signal to address already in use
13	ERQTS	Error in qtask request
14	ERTID	Task ID error
15	ERDCH	Data channel map full
16	ERMPR	System call parameter address error
21	ERSPC	File space exhausted
26	ERNAE	File name already exists
30	EREOF	End of file
32	ERWAD	Write access denied
37	ERPRP	Illegal priority
44	ERIBS	Device already in system
45	ERDNM	Illegal device code
46	ERSHP	Error on shared partition set
53	ERIVP	Invalid port number
57	ERIDP	Illegal destination port
65	ERACU	Attempt to close channel/device that is not open
73	ERNSA	Shared I/O request not to shared area
101	ERBOF	Positioning before beginning of file
102	ERPRV	Caller not privileged for this action
103	ERSIM	Simultaneous requests on same channel
104	ERIFT	Illegal file type
110	ERBLR	Attempt to block unblockable process
111	ERPRE	Invalid system call parameter
113	ERCIU	Channel in use
117	ERBSZ	Illegal block size for device
120	ERXMZ	Attempt to XMT illegal message
126	ERMIS	Disk and file system revision numbers do not match
127	ERIDD	Inconsistent DIB data
130	ERILD	Inconsistent LD
131	ERIDU	Incomplete LD
132	ERIDT	Illegal device name type
134	ERVIU	LD in use, cannot release
147	ERICN	Illegal channel
156	ERTXT	Message text longer than specified
164	ERSTS	Stack too small (?TASK)
165	ERTMT	Too many tasks requested (?TASK)
173	ERPNM	Illegal process name
224	ERNWN	No message waiting
242	EREAD	Execute access denied
243	ERFIX	Can not initialize LD, run FIXUP
244	ERFAD	File access denied
251	ERDIO	Attempt to issue MCA request with direct I/O in progress
252	ERLTK	Last task was killed
274	ERMRL	Memory release error
305	ERNAS	Process is not a server
306	ERCDE	Connection does not exist
307	ERCTF	Connection table full
77020	ERITD	Connection broken
77021	ERFTM	File/tape mismatch
400	ERCDU	Can not delete unexpired file
77000	ERVWP	Invalid address passed as system argument
77001	ERVBP	Invalid pointer passed as system call

Table B.1 Numeric assembled values of error codes

Appendix C

The ASCII Character Set



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

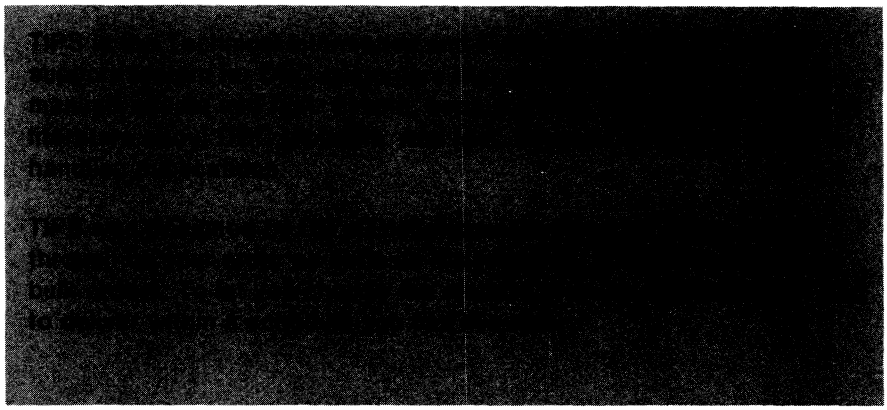
INTERNATIONAL OFFICES

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

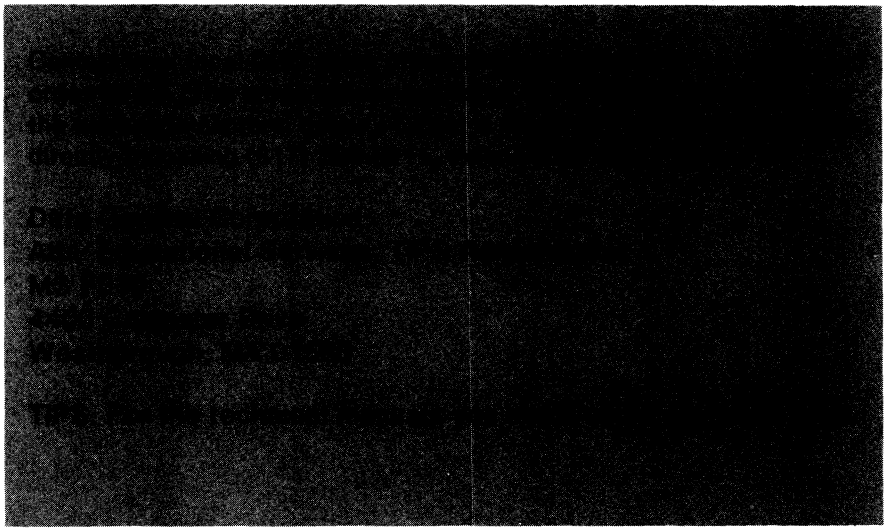
DataGeneral

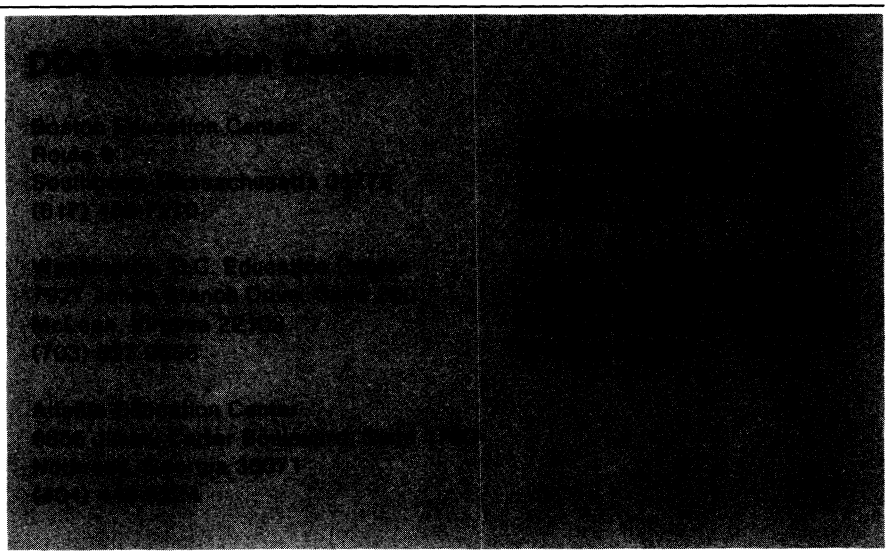
Ordering Technical Publications

Ordering Technical Publications



How to Get in Touch with TIPS





DataGeneral

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. 069-400028-00

Yes	No		
☐	☐	[redacted]	O You (can, cannot) find things easily. O Other: O Language (is, is not) appropriate. O Technical terms (are, are not) defined as needed.
		[redacted]	O Learning to use the equipment O To instruct a class. O As a reference O Other: O As an introduction to the product
☐	☐	[redacted]	O Visuals (are,are not) well designed. O Labels and captions (are,are not) clear. O Other:
☐	☐	[redacted]	
☐	☐	[redacted]	

Name: _____ Title: _____

Company: _____ Division: _____

Address: _____ City: _____

State: _____ Zip: _____ Telephone: _____ Date: _____

DG-06895

DataGeneral
Data General Corporation, Westboro, Massachusetts 01580

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