Advanced Diagnostic Executive System

ADES

Operator's Manual

ADES
Operator's Manual

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.

CEO, DASHER, DATAPREP, ECLIPSE, ENTERPRISE, INFOS, microNOVA, NOVA, PROXI, SUPERNOVA, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, TRENDVIEW, MANAP, SWAT, GENAP, and **PRESENT** are U.S. registered trademarks of Data General Corporation, and **AZ-TEXT, ECLIPSE MV/10000, DG/L, DG/XAP, GW/4000, GDC/1000, REV-UP, UNX/VS, XODIAC, DEFINE, SLATE, DESKTOP GENERATION, microECLIPSE, BusiPEN, BusiGEN,** and **BusiTEXT** are U.S. trademarks of Data General Corporation.

ADES Operator's Manual
014-000744-03
© Data General Corporation, 1983

Revision History:

Original Release - November 1982
First Revision - March 1983
Second Revision - July 1983
Third Revision - November 1983

Contents

Preface

This manual introduces Data General's Advanced Diagnostic Executive System (ADES). The manual explains how to load ADES, how to control the operating environment, and how to use ADES to trouble-shoot malfunctioning equipment. The trouble-shooting information, however, is general; for details on repairing a specific system, you should refer to the appropriate maintenance manual.

This manual assumes no previous familiarity with ADES, but does assume that you know:

- the basic parts of a computer system
- how to use an operating system to load and control other programs
- how to operate switches on the CPU's front panel (or a soft console equivalent) of the computer system on which you run ADES.

Structure of the Manual

This operator's manual has four sections, plus four appendixes. The first two sections, *Operations* and *Advanced Operations*, provide an overview of all the tasks you perform with ADES. These sections tell you when to use particular commands and utilities, but do not provide details on how to use them. For example, we briefly explain bringing up the operating system, controlling the operating environment, working with files and media, and trouble-shooting.

In contrast, the second two sections, *ADES CLI* and *ADES Utilities* provide complete details on all Command Line Interpreter (CLI) commands and all ADES utilities. Once you understand why you should use a CLI command or ADES utility, you should turn to these sections to find out how to do so. The commands and utilities are alphabetically arranged for easy reference. You will find that after using ADES a few times, you will be able to turn quickly to the descriptions that you need in these reference sections.

The *Appendixes* include additional information on topics such as trouble-shooting during power-up, panic codes, mnemonics, and model numbers. A glossary of terms is also included.

The sections are organized as follows:

Section 1: Operations

Chapter 1: What is ADES? describes ADES and gives an overview of some of the utilities which comprise ADES, the test programs supported, and general procedures for using the system.

Chapter 2: Bringing Up ADES takes you through the power-up sequence and briefly explains ADES CLI.

Chapter 3: The System Equipment Table describes the System Equipment Table and explains how you modify it with the Sizer programs. This chapter also tells you how to save the table for future use.

Chapter 4: Controlling the Operating Environment explains aspects of the operating environment that you can control, making particular mention of the Environment Control Word.

Chapter 5: The File System gives an overview of file types and of ways to manipulate files.

Chapter 6: Media describes different types of media and how to update, switch, and build them.

Chapter 7: Basic Trouble-Shooting explains how to test the system configuration, execute test programs, patch test programs, and bring down ADES.

Chapter 8: CLI Macros describes the usefulness of script files in trouble-shooting and tells you how to create them.

Section 2: Advanced Operations

Chapter 1: Advanced Trouble-Shooting explains how to control program execution through the switch register and string, and when to use the symbolic and octal debuggers.

Chapter 2: Conditional CLI explains the use of conditional CLI statements.

Section 3: ADES CLI

Chapter 1: Using the CLI explains CLI syntax and screen edit and control characters.

Chapter 2: CLI Commands includes a functional summary of commands, followed by a detailed description of each command, alphabetically arranged.

Section 4: ADES Utilities

Chapter 1: The Symbolic Debugger (ADEB) tells you how to execute the Debugger and summarizes its basic commands.

Chapter 2: The ADES Media Builder (AMB) describes the automatic and manual modes of operation for the Builder as well as specific instructions and commands used in manual mode.

Chapter 3: The Edit Utility describes the functions of Edit and the use of the program's commands. A short console session shows you how to use Edit.

Chapter 4: The Octal Debugging Tool (ODT) details the use of the ODT, the format of commands, and the commands themselves.

Chapter 5: The Script Builder (SCRBLD) describes the information sources of the SCRBLD, and how to execute it.

Chapter 6: Sizer Programs (ASIZE and MSIZE) outlines both sizer programs, automatic and manual.

Chapter 7: The Update Utility explains how to update a system disk without rebuilding it. The questions asked by the Utility are also given.

Chapter 8: ADES File Editor (FEDIT) tells you how to display and patch any file on the runtime media, and explains how to generate toggle files.

Chapter 9: Disk Edit Utility (DEDIT) tells you how to display and patch any write-enabled disk in the system.

Chapter 10: File Display Utility (DISPLAY) explains how to display the contents of a file in both numeric and ASCII format.

Chapter 11: DTR Form Print Utility (DTRFORMS) shows you how to print DTR forms.

Chapter 12: Contiguous Memory Test Utility (CMEMT) tells you how to verify that each 1 KW in memory exists.

The four appendixes are as follows:

Appendix A: Glossary
Appendix B: Power-Up Trouble-Shooting
Appendix C: Panic Codes
Appendix D: Mnemonic Descriptions

Typesetting Conventions

When we refer to commands, they appear in **UPPERCASE BOLD FACE** to distinguish them from other text. Any messages or prompts from ADES appear in this kind of face: ADES-CLI>. If the prompt includes an item in square brackets [ ], the item is a default. This means that if you just press NEW LINE, the system takes the item as your input.

All numbers in the book are octal, unless we indicate otherwise.

Command lines appear as follows:

type-CLI> **COMmand** *arg1 [arg2]*
[arg2 [arg3]]

where:

type-CLI> is the prompt that informs you that a particular utility is running and awaiting input from you. Here, type may be ADES, or EDIT, for example, ADES-CLI>.

COMmand is the name of the command, with the uppercase letters indicating how you can abbreviate the command. For example, **DELete** means that when you use the DELETE command you need only type in **DEL** (although you can type the entire command, if you wish).

argl	is an argument you must provide; for example, the name of a file. If you provide more than one argument, you must separate them with one or more spaces or one comma.
[ ]	are brackets indicating that the argument is optional. Never type in the brackets; they simply indicate that you have a choice. Notice that you can have options within options. Also, you may have a choice of arguments. In our examples, we will show choices in vertical lists; this means you choose one of them.
< >	are angle brackets indicating a list of arguments, separated by commas; you can choose one or more of these.
<↵>	is a terminator and consists of a NEW LINE, Carriage Return, or Line Feed. Press one of these keys when you see ↵

Section 1
Operations

Chapter 1
What is ADES?

ADES, the Advanced Diagnostic Executive System, lets you load and execute test programs. This operating system is intended for use on all Data General processors. See the current release notice for a complete list of equipment that applies.

ADES supports SEARCH (System Exerciser and Reliability Check) test programs, as well as its own set of test programs. ADES also includes a group of system programs.

Test Programs

The ADES test programs include:

- Diagnostic programs, which detect and isolate hard faults on subsystems and modules. These programs can also help isolate a problem in the failing circuit by providing functions such as looping on an error. Error reports indicate which Field Replaceable Unit (FRU) may be failing.
- Reliability programs, which detect hard, intermittent, and interactive faults on subsystems and at lower levels. These programs are used to verify operation in a simulated user environment, and their error reports provide information about the environment in which the error is detected.
- Exerciser programs, which detect and isolate hard, intermittent, and interactive faults in systems and distributed systems. Their error reports indicate either which subsystem may be failing, or which FRU.
- Verification programs, which prove that a product performs according to its functional specification.
- Formatter programs, which initialize magnetic media.
- Timing programs, which help calibrate equipment to meet timing specifications.
- Alignment programs, which help align the heads of magnetic recording devices.

System Programs

ADES includes a number of system programs:

- Bootstrapping programs for loading ADES into memory from either magnetic tape or disk
- Driver programs that allow ADES to communicate with peripherals
- The Command Line Interpreter (CLI), which serves as an interface between you and the operating system
- System utilities which are used by ADES programs
- Sizer programs for establishing the configuration of the system you are testing
- An edit utility for creating and editing files
- A script builder for creating macros to test equipment
- Debugger programs for tracking down software problems
- A media builder for creating ADES media on magnetic tape or disk
- An update utility for updating an ADES system disk.

General Procedures

Before you can use ADES, you must load it into memory from tape or disk. This process is known as *bootstrapping*. The programs responsible for bootstrapping run some quick diagnostic tests, load in and initialize the operating system, and transfer control to the Command Line Interpreter (CLI).

The CLI is the program you use to control your working environment, execute test programs, and run various utilities. Before starting any trouble-shooting, you need to do some preparation. For example, you would run sizer programs to set up the System Equipment Table. This table contains the configuration of the system you are testing. You might also use CLI commands to let you run off the system console of your choice, so that errors are reported as you want. Finally, if you are working off tape media, it is recommended that you build an ADES system on a disk for two reasons: to achieve response that is substantially faster, as well as to be able to create and save files. If the system is already built, you can switch to it or update it if necessary.

Once everything is set up as you want it, you can start trouble-shooting. If you just want to test the system configuration, you can use the RUN and ACCEPT macros. Otherwise, a variety of commands allow you to execute test programs. Additionally, ADES lets you use macro files to execute programs. These files contain a sequence of commands that allow you to control the order in which test programs are run. A Script Builder utility lets you generate macro (script) files automatically, whereas an Edit utility gives you more flexibility in the length and contents of macro files. The Edit utility also lets you patch any test programs that need updating and maintain a file of notes to yourself.

ADES also includes more advanced trouble-shooting facilities. For example, you can use either a symbolic or octal debugger to see where a test program is failing. You can also control the actions of a program via a Switch Register (SWREG) or a string buffer (STRING). Chapter 1 of Section 2, "Advanced Operations," and Section 4, "ADES Utilities," explain these trouble-shooting facilities in greater detail.

Chapter 2
Bringing Up ADES

As mentioned previously, bootstrapping is the initial process of loading ADES from tape or disk into memory. The tape or disk containing ADES is the ADES system media. When you load it into memory and it executes, it is known as the runtime media.

The program responsible for bootstrapping includes a test that checks the basic functions of the system hardware (excluding peripherals) during bootstrapping.

After bootstrapping, some initialization programs complete the process of loading the system. When successfully loaded, ADES issues the ADES-CLI> prompt, indicating that you can start to enter CLI commands.

Bootstrapping

To bootstrap ADES, you should follow the instructions in the documentation for your hardware system. This procedure varies from system to system. If bootstrapping is successful and the system hardware passes the test given by the ADES boostrap program, you will get the message:

```
FILENAME [ADES]?
```

If the test fails, the CPU is halted before the words are completely displayed. If the message does not appear within 30 seconds, some problem has been detected. See Appendix B for more detailed information.

Initializing

If the FILENAME [ADES]? prompt appears, you now have the choice of loading and executing either ADES or a specialized standalone program. Generally, standalone programs are used when the system lacks sufficient memory; ADES requires a minimum of 16 kilowords (KW).

To load a standalone program, type in its name, then press NEW LINE. To load ADES, just press NEW LINE. (Remember that the square brackets around the word ADES indicate that ADES is the default answer to the question.) If you are unsuccessful at loading ADES, refer to the last portion of this chapter, *If You Cannot Load ADES*.

When you load ADES, a program called the Initialization Monitor takes over. This program initializes some system variables. From this point, you may get a fatal ADES error, also known as a system panic, should the operating system encounter a problem. For details on system panics, see Appendix C.

If the Initialization Monitor encounters no problems, the System Initialization program takes over and concludes the initialization sequence. You then get a message about the copyright, CPU, and the size of memory, as shown at the top of Figure 2.1.

```
FILENAME [ADES]?

COPYRIGHT (C) DATA GENERAL CORPORATION, 1983. ALL RIGHTS RESERVED.
LICENSED MATERIAL, PROPERTY OF DATA GENERAL.

CPU IS XXXX.
MEMORY SIZE IS XXXX KW.

RUN AUTOSIZER (Y,N) [Y]?
REPORTING LEVEL (?,0,1) [0]?

SIZING SYSTEM CONFIGURATION.  (These messages appear if you choose
SIZING COMPLETE.               reporting level 0.)

ERCC     02F                  (Mnemonics and device codes such as
MMPU1    03F                   these are printed if you choose
RTC      14F 54                reporting level 1.)
PIT      43
PTP      13  53
FPU      76F
MTA      22F 62

SECONDARY OUTPUT DEVICE:  xx, mmmm

**** ADVANCED DIAGNOSTIC EXECUTIVE SYSTEM   REV 0.0 ***

ENTER "HELP" FOR HELP.

ADES-CLI>
```

Figure 2-1. Loading and Initializing Sequence

Sizing Peripherals

Next, you will be asked whether you want to run the Auto Sizer program. This program sizes (i.e., identifies) all the peripherals it can find and adds the information to the System Equipment Table, which contains the configuration of the system.

Run the Auto Sizer if:

- You are loading ADES from tape or have not previously saved a file called EQUIPO.
- You want to check quickly whether all the peripherals are working. If the Auto Sizer cannot find a peripheral, that peripheral may either be malfunctioning or offline.

If you choose to run the Auto Sizer, then you must select the appropriate reporting level (0 or 1). If you choose a reporting level of 0, two messages will be printed out: one tells you sizing is being done; the other that it is completed (see Figure 2.1). If you indicate reporting level 1, then the Auto Sizer prints out a list of device codes and mnemonics for the devices the Auto Sizer looks for. If the program finds a device, it adds an F after the code. For example, MTA 22F 62 means that the Auto Sizer found a tape drive on device code 22, but not one on 62. You can see this more clearly in Figure 2.1. (Typing in a question mark means that the listed choices will be explained.)

NOTE: For tape and disk drives, an F after the device code means that at least a controller has been found; there may or may not be online units attached to the controller.

The default reporting level is 0. If you type in a question mark followed by NEW LINE, or an illegal response, the following explanation is printed:

```
0 = SUPPRESS AUTOSIZE OUTPUT.
1 = PRINT MNEMONICS AND DEVICE CODES AS THEY ARE SIZED.
```

If, however, you are loading ADES from disk and have previously built a System Equipment Table and saved it in a file called EQUIP0, you need not run the Auto Sizer. In this case, the System Initializer looks for EQUIP0 to use as the System Equipment Table. If the program finds EQUIP0, you get a message telling you so. Otherwise, you get a warning that the equipment table is empty. If you do get a warning, you must build a System Equipment Table, as explained in Section 1, Chapter 3, "The System Equipment Table."

Once the System Equipment Table is set up, the System Initializer program looks for a description of the system console in the SET. If it does not find an entry in the SET describing the system console, the program assumes that it is a hardcopy device, model number 6040. To change the system console to a video terminal, follow the procedures for updating the System Equipment Table in Section 1, Chapter 3, "The System Equipment Table," after you get the CLI prompt.

The System Initializer now tries to find the secondary output device in the System Equipment Table. If the program finds it, you get a message, as in Figure 2.1. Otherwise, you get a warning message. If this happens and you are planning to use a printer as a secondary output device, first make sure that it is online. Then, after you get the CLI prompt, run the Auto Sizer so that ADES knows of the printer's existence. Finally, use the **SECONDARY** and **ENVIRONMENT** commands, as explained in Section 1, Chapter 4, "Controlling the Operating Environment."

As a final step, the RISE macro is executed if it exists on the runtime media. This macro exists only if you had previously created it. You define RISE as you would define any other macro; it is particularly useful if you know just what sequence of commands you want to execute when you first bring ADES up. The RISE macro ensures that this sequence is carried out automatically. See Section 1, Chapter 8, "CLI Macros" for a complete discussion of macros.

Then the CLI prompt appears:

```
ADES-CLI>
```

What is the CLI?

The ADES Command Line Interpreter, or CLI, lets you interact with the operating system. The CLI lets you control a number of system variables, such as the system console, the radix used by the operating system, and the way errors are reported by test programs. The CLI also lets you manipulate the many kinds of files that ADES supports. In addition, CLI commands let you execute system utilities (such as the Edit utility or Script Builder), as well as test programs.

You can use the CLI in two modes: manual and script. In manual mode, the CLI processes commands that you enter in response to the CLI prompt, ADES-CLI>. In script mode, the CLI processes commands that it finds in a script, or macro, file that you previously created. Macros can save a lot of time when you frequently execute a sequence of commands. Instead of typing in the sequence each time, you need only provide the name of the macro file in response to the CLI prompt; the CLI then executes all the commands in the file. For a detailed discussion of macros, see Section 1, Chapter 8, "CLI Macros."

To enter a CLI command, you must type in the command name. Some commands then require one or more pieces of information, known as arguments. If you are providing an argument, leave a space or type a comma between the command and the argument. Finally, to transmit the command, press the NEW LINE, Carriage Return, or Line Feed key. For details of entering commands, see Section 3, "ADES CLI."

If You Need Help

ADES CLI includes a **HELP** command, plus a tutorial mode, should you need some help in using the operating system.

The **HELP** command gives you information about CLI commands, related topics, system programs, and test programs. You can use HELP whenever you are using manual mode under ADES CLI.

To see a list of items about which you can obtain helpful information, just type:

```
ADES-CLI> HELP
```

You can get information on all CLI commands, on programs residing on the runtime media, and on topics.

Topics described by the HELP facility include:

CCHARS	Control character and keystroke commands
CFLIST	ADES media builder custom files lists
CLI	All ADES CLI commands
DEVICE_STATUS	Device status words
FTYPES	ADES file types
MACROS	ADES standard macros
MNEMONICS	ADES mnemonics and model numbers
PANICS	ADES panic codes
SEDIT	Screen edit utility
SYNTAX	Syntax conventions used in help files
UTILITIES	List of available ADES utilities
XREF	Diagnostic programs cross reference listing.

To obtain information about a specific item, type:

```
ADES-CLI> HELP item
```

where *item* is the subject about which you would like some information. For example, the item may be the name of a program, a command, or a topic.

ADES CLI also provides tutorial help whenever a system flag called OPERATOR is on and you omit arguments to a command which requires them. The CLI then steps you through a tutorial session so that you understand the arguments you must provide. When you bring ADES up, OPERATOR is automatically on, so you will get tutorial help unless you turn this flag off (see Section 1, Chapter 4, "Controlling the Operating Environment").

If You Cannot Load ADES

If you fail to load ADES the first time, any number of system failures may be the cause. First check whether:

- The CPU power is on
- The CPU is unlocked
- The system operator's console is online
- The system media is online and at BOT (if tape) or recalibrated (if disk)
- All the cables are in good condition and plugged in properly.

If the answer to any of these is no, correct the situation and try loading ADES again.

If only part (or no part) of the words FILENAME [ADES]? appears, see Appendix B for further information.

If FILENAME [ADES] appears, but you then encounter a system panic during initialization, go to Appendix C for information.

If, in either case, you do not suspect the hardware, try rebuilding the media (see Section 1, Chapter 6, "Media").

Chapter 3
The System Equipment Table

The System Equipment Table (SET) is a table in memory that lists all the peripherals and options in the system. The SET includes information about the model numbers, device codes, and unit numbers of peripherals, as well as the CPU type and memory size. The table also lists the type of the media and its write protection status. For disks, the dimensions are included. This kind of information is crucial to the diagnostic, reliability, and exerciser programs for peripherals.

To set up the System Equipment Table, you can use the Auto Sizer program, the Manual Sizer, or both. The Auto Sizer cannot size all peripherals (notably communications ones) and may be unable to size malfunctioning peripherals correctly. Also, the program makes a number of assumptions about device-dependent information. Such information could be line characteristics information on an Asynchronous Line Multiplexor (ALM). So, generally, you will run both sizer programs to set up the System Equipment Table completely.

Sizer Programs

You can run the Auto Sizer either when bringing up ADES (refer back to the chapter on "Bringing up ADES") or any time after you have a CLI prompt. The Auto Sizer (ASIZE) automatically sizes all the peripherals that it finds and adds the information to the System Equipment Table. As you may recall, ASIZE produces a list of device codes and mnemonics. For any device that it finds, the Auto Sizer adds an F after the device code. Examine this list carefully to see if the program is omitting any devices. (You can also use the **EQUIPMENT** command to check the contents of the SET.) If any peripherals are missing, use the Manual Sizer to enter information about them.

The Manual Sizer lets you add information manually to the SET about any peripherals in the system. Generally, as we have pointed out, you will use the Manual Sizer (MSIZE) to add peripherals that ASIZE has omitted. For example, if the System Initializer did not find a secondary output device, you should update the System Equipment Table. Similarly, if ASIZE assumed the system console was a hardcopy 6040 (see Chapter 2), you may want to redefine it as a video console using MSIZE.

For details on both sizer programs, refer to Section 4, Chapter 6, "Sizer Programs (ASIZE and MSIZE)."

Saving, Updating, and Clearing the SET

Since the SET is a table in memory, when you bring down ADES the SET is gone. For this reason, once you have run the sizer programs, consider using the CLI **EQUIPMENT** command, which allows you to save SET information in a disk file called EQUIP0. This file saves you running the sizer programs each time you bring up ADES. If you have saved the information from SET in EQUIP0 (see Chapter 2), you can avoid running the Auto Sizer altogether and have the System Initializer set up the SET with information from EQUIP0.

In addition to letting you save the SET, the **EQUIPMENT** command serves other useful functions. It lets you:

- Display the contents of the System Equipment Table
- Save the SET in a file of your choice (that is, a file other than EQUIP0)
- Replace the contents of SET with either the contents of EQUIP0 or a file of your choice which contains a SET
- Clear the contents of the SET (but preserve any versions saved on disk).

Three other CLI commands are useful in working with the System Equipment Table: **PROTECT, UNPROTECT**, and **SCRATCH**. **PROTECT** and **UNPROTECT** set up and eliminate software write-protection, respectively, for a disk or tape. This information is stored in the SET. **SCRATCH** is used to change the media type of any disk or tape to SCRATCH MEDIA.

For details of CLI commands, see Section 3, "ADES CLI."

NOTE: When you bring up ADES, unknown media are classified as write-protected unless bit 1 in the System Status Word (of the Primary Label Block) is 1, in which case unknown media are classified as scratch.

Chapter 4 Controlling the Operating Environment

Before you start running any programs or utilities, we suggest that you make sure your operating environment is set up as you want. A number of CLI commands let you display or set system variables, such as:

- Output devices (system console and secondary device)
- System identification
- System defaults
- Error class for CLI commands
- Radix
- Environment Control Word (controls many aspects of the operating environment).

Output Devices

If you wish to switch the system console to another device, use the **CONSOLE** command; to switch and enable the secondary output device, use the **SECONDARY** and **ENVIRONMENT** commands (change the SECONDARY flag), respectively. You can also use the **CONSOLE** and **SECONDARY** commands to find out what the system console and secondary output devices are.

System Identification

The CLI gives you the option of establishing an 80-character system identification message. This is very useful for identifying the runtime media. For example, your message could include information about the date and time when the media was built or the model number of the system. To display or set up a message, use the **SYSID** command.

NOTE: You cannot change the system identification when the runtime media is a tape.

System Defaults

You can set certain defaults with the **DEFAULT** command. For example, you can determine how many passes a test program makes, if the pass count is not specified on the **XEQ** command line (the initial setting is 1).

Error Class

Initially, the error class for CLI commands is set to ERROR. This means that if the CLI encounters a problem, it does not execute the rest of a command line or a script file. All error messages begin with the word ERROR.

The **ECLASS** command gives you the option of changing the error class to WARNING or IGNORE. If you set it to WARNING and the CLI encounters a problem, it issues a warning but continues to execute the rest of a command line or macro. If you set it to IGNORE and the CLI encounters a problem, it ignores it and issues no message.

You can find out the status of the error class with either the **ECLASS** command or the **STATUS** command.

Radix

ADES assumes a radix of eight (octal), unless you indicate otherwise. If you wish to change the radix of all numbers to ten (decimal) or sixteen (hexadecimal), use the **RADIX** command. This command also lets you change the radix back to octal. To change the radix of one number, you can use a radix indicator; follow the number with K for octal, H for hexadecimal, or . for decimal.

Environment Control Word

The Environment Control Word is a word in memory that lets you control many aspects of your operating environment. Each bit of the word represents a flag which you can turn on or off, depending on the effect you want. When you bring ADES up, these flags are set as shown in Table 4.1.

Four commands--**ENVIRONMENT, LOG, NOTIFY**, and **OPERATOR**--let you define the flags in the Environment Control Word. Table 4.1 shows which commands you should use to change the effect of any flag. The same commands let you see the status of the flags that they control. You can also complement (that is, change to the opposite state) the condition of any flag by using single or double keystrokes. These keystrokes provide a short-cut to changing the operating environment.

Table 4-1. Environment Control Word Flags

Flag Name	Effect	Initial Value	Command
LOOP	Loop when an error is detected	Yes	ENVIRONMENT
CONSOLE	Print output on system console	Yes	ENVIRONMENT
PERCENT	Print percentage failure at end of tests	No	ENVIRONMENT
PASSES	Notify of start of passes	Yes	NOTIFY
SECONDARY	Enable output to secondary output device	No	ENVIRONMENT
PAUSE	Execute the Octal Debugging Tool (ODT) if error is detected	No	ENVIRONMENT
TESTS	Notify of start of tests	No	NOTIFY
REPORTALL	Print all errors encountered	No	ENVIRONMENT
TERMINATE	Terminate program if error is encountered	No	ENVIRONMENT
LOGGING	Enable console logging	No	LOG
MODULES	Notify of start of modules	No	NOTIFY
VERIFY	Run only quick verification tests	No	ENVIRONMENT
PAGE_MODE	Set page mode (that is, suspend output every 23 lines or when a formfeed is encountered)	No	ENVIROMENT
ABORT	Terminate current program and abort current script if an error is detected	No	ENVIRONMENT
OPERATOR	Operator present	Yes	OPERATOR
DCHANNEL	Initiate background data channel activity on the second and subsequent passes of diagnostic test programs via the Data Channel Exerciser Utility	No	ENVIRONMENT

Table 4-2. Single Keystroke Commands

Command	Flag
1	LOOP
2	CONSOLE
3	PERCENT
4	PASSES
5	SECONDARY
6	PAUSE
7	TESTS
8	REPORTALL
9	TERMINATE
A	ENVIRONMENT
B	MODULES
C	VERIFY
D	PAGE_MODE
E	ABORT
F	OPERATOR
G	ENABLE DATA CHANNEL ACTIVITY

Single Keystroke Commands

You can use single keystroke commands only while a program is running or while something is being printed (that is, when you have no prompt of any kind) to complement the meaning of any flag in the Environment Control Word. Never use these commands when you have a CLI prompt or when a program is waiting for input; the commands would be interpreted literally. If you have a prompt, use a double keystroke command (see below).

Single keystroke commands are single digits or alpha characters. Each command corresponds to a flag in the Environment Control Word. For example, to complement the value of the REPORTALL flag, you would type 8. See Table 4.2.

Single keystroke commands are printed as follows:

- On hardcopy terminals (model 6040) and DGC Displays (model 6012), as # plus the character typed
- On uppercase LCDs (model 6052), the # plus the character are blinked
- On upper/lowercase displays (models 6053, 6106, 6130), graphics displays (model 6150), and color displays (model 5220), the # plus the character are dimmed.

Double Keystroke Commands

You can use double keystroke commands whenever you have a CLI prompt or a program is awaiting input. As we said above, if you were to enter a single keystroke command, the program would interpret it literally. To prevent this from happening, you must precede a single keystroke command with a CTRL-P, thus making it a double keystroke command. For example, to complement the value of the OPERATOR flag, type CTRL-P F.

NOTE: Uppercase or lowercase may be used for the letter commands A-G listed in Table 4.2.

WARNING: DTOS programs may not recognize Keystroke commands.

Double keystroke commands are printed as follows:

- On hardcopy terminals (model 6040) and DGC Displays (model 6012), as ^P# plus the character typed
- On uppercase LCDs (model 6052), the ^P# plus the character are blinked
- On upper/lowercase displays (models 6053, 6106, 6130), graphics displays (model 6150), and color displays (model 5220), the ^P# plus the character are dimmed.

Chapter 5
The File System

ADES has a flat file system, which means that all files are on one level. Once the operating system is running, you have access to any file without changing your working environment in any way.

File Types

ADES has fourteen types of files, summarized in Table 5.1.

Table 5-1. File Types

Type/Extension	Meaning
DTOS	DTOS program (cannot use any ADES control characters).
RSYS	System program, which remains memory resident once it is invoked.
PRG	Program file which runs under ADES.
OLF	Overlay file (contains executable code).
UDF	File with unstructured data (raw numeric data) used by a user program.
SDF	System data file, which contains unstructured data used by an ADES utility.
SNRU	System nonresident utility.
DRVR	Driver file, which performs I/O operations to peripherals supported by ADES.
SRCE	Source file, or macro file, containing CLI commands. The extension .CLI may be substituted for .SRCE in ADES CLI commands. Note, however, that ADES will confirm files with the .CLI extension as .SRCE files.
TOGL	Toggle file, which contains instructions to perform a specific function.
HELP	HELP file, which explains some aspect of ADES or a test program. SRCH SEARCH file (System Exerciser and Reliability Check).
SOP	Standalone program.
TXT	ASCII text file.

As you can see, ADES includes some system files, such as RSYS, SNRU, SDF, and DRVR. You will probably be aware of the existence of these files, but not of how the files are being used.

System files are required by ADES in order to run; user files are not. Generally, user files include all help files, test programs, and their overlay and data files.

As you run test programs under ADES, you may be working with PRG, OLF, DTOS, and SRCH files; all four contain executable code. When you start using macros as a short-cut to executing a sequence of CLI commands and test programs, you will be working with SRCE files. You can create source files with either the Script Builder or the Edit utility (for more details see Section 1, Chapter 8, "CLI Macros").

You can also use the Edit utility to create two other kinds of files: TOGL and TXT. Toggle files contain instructions to save you from toggling switches. For example, you could have a toggle file to set up a seeking/recalibrating loop for disks. Text files are simply ASCII text files that can contain any kind of textual information; for example, a message or a memo.

Filenames

Files are identified by a name and a type. The names of files may be up to 14 characters long and may include any character except a period (.) or a space. When you name a file, follow the name with its type. This extension consists of a period followed by one of the types listed in Table 5.1. The format is:

filename.filetype

For example, RENEE.TXT is a text file, but ANNA_22.SRCE is a source file. A filename followed by an extension is also known as a pathname.

Although not all CLI commands require an extension (that is, the file type), we suggest you always provide one. As a result, you avoid misidentification.

When naming files, you must identify them uniquely. You can, however, have several files with the same name, but different extensions. For example, you could have files called ANNA.PRG, ANNA.OLF, ANNA.TXT, and ANNA.SRCE.

In working with files, you must refer to them by name. Three CLI commands **FILESTATUS, DELETE,** and **PERMANANCE**) and one ADES utility (ADES Media Builder), however, allow you to take a short-cut by using template characters.

Templates

ADES includes two template characters: * and +. These characters serve as a shortcut in referring to filenames.

The * template tells the system to look for any file that includes the characters you indicate, plus any other single character. For example, if you had the files listed in Figure 5.1 and you wanted to find any file beginning with the characters ACT and followed by any other single character, you would use the * template as follows:

ACT*

Given the selection in Figure 5.1, the system would find:

ACT1.SNRU ACT2.TXT ACTB.OLF

Notice that the system finds any file beginning with ACT and followed by any single character, regardless of its filetype extension.

```
ACT1.SNRU     ACT11.SRCE
FOO3.TOGL     ACTFOO2.TXT
ACT2.TXT      ACT2356.TOGL
2234.UDF      1700.UDF
ACTB.OLF      PROG23.PRG
```

Figure 5-1. A Selection of Files

The + template tells the system to look for any file which includes the characters you indicate, plus any number of other characters. For example, to find any file beginning with the characters ACT and followed by any other characters, you would type:

ACT+

This time, the system would respond with:

```
ACT1.SNRU     ACT11.SRCE     ACTFOO2.TXT
ACT2.TXT      ACTB.OLF       ACT2356.TOGL
```

You can also use either template character to find files which end with the characters you indicate. For example, to find all files ending in 2, you would type:

+2

Here, because you are looking for all files ending in a certain character, you place the + first. The system would now respond with:

ACT2.TXT ACTFOO2.TXT

Lastly, you can use templates in the middle of characters you want to match. For example, if you type:

A+1

with the selection of files in Figure 5.1, the system would find ACT1.SNRU and ACT11.SRCE.

You can even combine templates as follows:

*CT+

In this case, all files with one character before CT and with any number of characters after CT will be found. With the files in Figure 5.1, you would get:

ACT1.SNRU	ACT2.TXT	ACTB.OLF
ACT11.SRCE	CTFOO2.TXT	ACT2356.TOGL

Manipulating Files

Several CLI commands allow you to work with the file system. To find out what files are on the runtime media, you can use the **FILESTATUS** command. This command lets you list all files, or just those with a certain extension, and/or those whose names have certain characters. You select categories of files by using template characters, as we explained above.

If you are working off a disk, you can find out how much disk space you are using and how much is free with the **SPACE** command.

If you want to see what certain files contain, you can display them with the **TYPE** command. This command, however, applies only to files containing ASCII text (that is, SRCE and TXT files).

To make a copy of any file, use the **COPY** command. To add one macro file (SRCE) to another, use the **APPEND** command.

You can also change the name of any file with the **RENAME** command and you can delete any non-permanent file with the **DELETE** command. To make a file either permanent or non-permanent, use the **PERMANENCE** command.

NOTE: The ADES Media Builder and the Update utility automatically create permanent files. The Edit utility creates non-permanent files.

Two additional commands let you manipulate files: **TLOAD** and **TDUMP**. **TLOAD** loads a file from the specified file number on the tape drive indicated and **TDUMP** dumps a file to the drive indicated.

Details on these commands are in Section 3, "ADES CLI."

Chapter 6
Media

The runtime media is the media from which ADES is currently executing; a system media or ADES media is any media (disk or tape) which contains ADES software. However, at any point a system media can become the runtime media.

You can find out what the runtime media is with either the **MEDIA** or **STATUS** commands. You can also use the **MEDIA** command to make a system media the runtime media.

Four other types of media exist: scratch, write-protected, console log, and update. A scratch media is any device media available for testing; the data on scratch media is vulnerable. In contrast, as its name suggests, write-protected media cannot be written to. Generally, any customer disk packs with DGC software are write-protected. When ADES comes up, unknown media are classified as write-protected (unless bit 1 in the System Status Word is 1, in which case unknown media are classified as scratch).

WARNING: Software write-protection restrictions do not apply to DTOS programs. To protect magnetic media, make sure the device is offline before running DTOS programs.

A console log media is a tape which holds the console log. For details see the description of the **LOG** command in Section 3. Finally, the update media contains update files (see later in this chapter).

At times you may want to build a new system media. Running ADES off tape has its disadvantages: it is slow; you lose the ability to create and save files; and you cannot run a full set of test programs in an automatic test sequence. You can eliminate these restrictions by building, and then running from disk. If you need to distribute a system easily, however, you may want to build it on tape. The ADES Media Builder (AMB) lets you do both.

Building Media

The ADES Media Builder can create a system only on scratch media (a media that is not write-protected) or on an ADES media (which already contains ADES software). You can check on the status of the media with the **EQUIPMENT** command. If the media is write-protected, change its status either by using the **UNPROTECT** or **SCRATCH** commands, or by running the Manual Sizer (MSIZE) and changing the status of the device to scratch.

The ADES Media Builder (AMB) operates in two modes: automatic and manual. Automatic mode is useful if you want to include all the files which make up the runtime media in the media you are building. In automatic mode, the Media Builder builds a system on the first scratch media it finds in the System Equipment Table that can build a media of the model number specified. (The table is ordered by device code and unit number.) The Builder copies all the files from the runtime media to the scratch media. If you want to control which device the Media Builder uses, you must make sure that all the devices ahead of it in the SET have write protection.

In contrast, manual mode is used if you want to select only a subset of the enabled files for the new media, if you want to build to a scratch device other than the first applicable scratch device listed in the SET, or if you want to build more than one media at a time.

In either mode, you can customize the new media. The files transferred to the build media can be customized in two ways: by selecting files and/or programs based on the contents of a System Equipment Table, or by selecting files contained in a specified TXT file.

For complete details on both modes of the Media Builder, see Section 4, Chapter 1, "ADES Media Builder (AMB)."

Updating Media

To allow you to update an ADES system disk without rebuilding it, ADES includes an Update utility. This utility uses either an update tape or system disk to update the runtime media. When running the utility you have the option of copying some or all of the files from the update media to the runtime media.

The update media is distributed by Data General Corporation, but users also have the option of creating it with either AOS or RDOS commands. This is especially beneficial for users who write their own diagnostics and wish to include their own files in the update.

For details on the utility, refer to Section 4, Chapter 7, "The Update Utility."

Chapter 7
Basic Trouble-Shooting

When trouble-shooting a system you generally do the following:

1. Try to evaluate the situation
2. Run test programs to isolate the failing device
3. Repair the system
4. Verify the repair.

Here we will discuss steps 1, 2, and 4; for step 3, refer to the maintenance manuals for the system.

Preliminary Evaluation

Whenever possible, talk to the person who was working with the system when the failure occurred. Also, carefully examine any printouts that document the machine's state when it failed.

After gathering all the available information, you should have answers to the following:

- Is the failure intermittent or a one-time event?
- Is the system still operational?
- Is the error always the same?
- Do the symptoms form a consistent picture?
- Has any hardware or software been changed recently?
- Has the system ever had similar problems? What were the conclusions at the time?
- What does the user think the problem is?

Your answers to these questions let you form an initial conclusion about the symptoms:

1. They do not provide a clue to the source of the problem
2. They point to a general area of the system, but not to a specific device
3. They point to a specific device.

Running Test Programs

Once you have an initial conclusion about the problem, you can start running test programs. What you run depends on your initial diagnosis.

If you have no clues about the source of the problem, you are best off using either the RUN or ACCEPT macros. These macros execute a standard group of diagnostic programs on the equipment currently in the System Equipment Table. Be sure, therefore, that the SET is up to date. Also, be sure that any disk or tape drives you are testing are not write-protected.

The RUN macro tests each device and function and ensures that each test program goes through three passes. The ACCEPT macro also tests each device, but has each program make only one pass.

If these tests show nothing, you can then try SEARCH. If these still fail to show anything, you should call Corporate Technical Support.

If, on the other hand, you know which device or part of the system is at fault, you should execute the test programs geared for that device. For details on test programs, see the individual HELP files.

To execute ADES programs or TOGL files, use the **XEQ** command. The **XEQ** command executes one program at a time. If you wish to execute several programs in a row, you have two options. You could either set up a script (or macro) file containing a series of commands, or you could use a multiple command line. In a multiple command line, you string the programs together, each separated by a semicolon. (Turn to the next chapter for details on script files.)

If you use a multiple command line, you must be sure that all the names of the programs you want to run will fit on one line (68 characters). If they don't, you must write a script. If you want to execute a sequence many times, you should also write a script.

You can also execute ADES programs with the CLI **LOAD** and **START** commands, but this method is not recommended because **LOAD** does not update certain system variables. Therefore, you will get erroneous information if these variables are used.

You can also load and optionally start a program on a slave processor (for example, a DCU) with the **SLAVE** command.

Depending on the errors the test programs report, follow the procedures in the maintenance manual for fixing your system.

Patching Programs

If you receive any patches (fixes or additions) to test programs, you should update the program files with the Edit utility (refer to Section 4).

Verifying the Repair

Once you have found the problem in your system and repaired it, verify the repair as follows:

1. Re-run the programs that isolated the problem. If the programs run without errors, the problem has been solved. If the failure recurs, restore any FRUs that you changed and start the trouble-shooting process again.
2. Run a complete set of system exerciser programs to make sure the system is working properly. If the exercisers disclose a new problem, start trouble-shooting again.

Bringing Down ADES

Once you have finished running diagnostics, you must bring down ADES. To do so, use either the **BYE** or **BOOT** commands. Both commands shut down the system in an orderly fashion, but **BOOT** also lets you reboot the device you choose, such as your normal operating system (like AOS). **BOOT** saves you from using **BYE** and then bringing up the operating system manually. When you use the **BOOT** command, you specify the device you want booted.

Chapter 8
CLI Macros

As we have mentioned, macros provide a short-cut in executing sequences of diagnostic programs, CLI commands, or both. Instead of typing in command lines one by one, you can enter them all in a macro file and then run the script. Macros are useful either when you wish to execute the same sequence of commands frequently or when you have too many commands to fit on a multiple command line. If you write a macro for such lengthy command sequences, then you don't have to wait for one test program to finish before typing in the execution line for the next. We will provide examples of macros as we continue.

WARNING: ADES CLI commands are valid in script mode, but Edit commands are not.

To write a macro, you create a text file known as a source file. This file contains a series of CLI commands. To execute the commands in the macro file, all you need to do is type in the name of the file.

NOTE: CLI macros can also be written under AOS, and can be included in the ADES media by inserting the name of the CLI source file in your input list.

Nesting Macros

You can avoid long and perhaps repetitious macros by having one macro call another. For example, you may have a macro XYZ that performs a common operation which is part of a number of different procedures. The macro for the larger procedure could include a command line with the name XYZ on it. When the CLI came to this line, it would execute the contents of macro file XYZ.

Note that you do not have to use the .SRCE extension unless the name of the macro is also the name of a CLI command, or its abbreviation. For instance, if you had a macro called TYPE, you would have to use TYPE.SRCE so that the macro is executed rather than the CLI **TYPE** command.

The process of referring to other macros inside a macro file is known as nesting. When a nested macro finishes, the CLI returns to the next line in the original macro. You can have up to three levels of nesting.

Chaining Macros

In addition to nesting macros, you can link them with the **CHAIN** command. If you chain to another macro, when that macro finishes, you do not return to the macro that contained the **CHAIN** command, but to the macro or CLI level above it. This gives you the choice of executing commands in a pattern that differs from nesting. An example of the nesting and chaining macros is given in Figure 8.1.

```
ADES-CLI>  TYPE MACROA.SRCE
WRITE THE BIG
MACROB
WRITE LAZY DOG.

ADES-CLI>  TYPE MACROB.SRCE
WRITE BROWN
MACROC
WRITE OVER THE

ADES-CLI>  TYPE MACROC.SRCE
WRITE FOX
CHAIN MACROD
WRITE NOTE THAT THIS LINE IS NOT PRINTED.

ADES-CLI>  TYPE MACROD.SRCE
WRITE JUMPED

ADES-CLI>  MACROA

THE BIG
BROWN
FOX
JUMPED
OVER THE
LAZY DOG.
```

Figure 8-1. Nesting and Chaining Macros

Macro Arguments

All arguments supplied on the command line are passed to the macro. You can supply arguments through the ADES Dummy Argument Facility. Dummy arguments are enclosed in percent signs, and have effect only in the body of the macro. When ADES invokes a macro, it generates and executes a temporary SRCE? file, with all the dummy arguments replaced with the requested macro arguments.

ADES supports three dummy argument formats:

1. %N% - Insert argument N.
2. %N-% - Insert arguments N through last.
3. %N-M% - Insert arguments N through M.

Argument 0 is the name of the macro. If a reference is made to a non-existent argument, ADES ignores it. If an invalid format is used, the requested macro will not be run, and the error INVALID DUMMY ARGUMENT FORMAT IN MACRO results. Note that macro argument numbers are parsed in decimal.

Example:

```
ADES-CLI> TYPE SAMPLE1.SRCE
WR HELLO<54> %1% %2%
WR I SEE YOU ARE %3-4% OLD, AND YOU ARE A %5-%.
WR GOOD FOR YOU!! %8-%

ADES-CLI> SAMPLE1 JOHN SMITH,29 YEARS,SENIOR ACCOUNT EXECUTIVE
HELLO, JOHN SMITH.
I SEE YOU ARE 29 YEARS OLD, AND YOU ARE A SENIOR ACCOUNT EXECUTIVE.
GOOD FOR YOU!!

ADES-CLI>
```

Creating Macros

You can create macros in three ways:

1. By writing them under AOS and including them in the ADES media
2. By using the ADES Script Builder
3. By using the Edit utility.

ADES Script Builder

The ADES Script Builder (SCRBLD) is used to generate a list of programs that run on the current CPU, test equipment that is listed in the SET, and that can be executed without asking any questions. On a tape, these programs are in the form of a memory-resident script that is executed and lost when the script finishes. On a disk, however, you have a choice. The programs can be in the form of a memory-resident script, or they can be in the form of a source file that contains a series of execute commands which can be saved on the runtime disk for later use.

To create either script, SCRBLD uses information from the System Equipment Table, from a cross-reference table which associates test programs with different devices, and from the directory structure which lists all the test programs on the runtime media.

For details on the Script Builder, refer to Chapter 5 in Section 4.

The Edit Utility

The Edit utility lets you create and edit script files (in addition to other file types). With this utility you can insert, delete, modify, and display text. You can run a one-page macro right from the Edit utility. Finally, you have the option of saving files on the runtime media.

With Edit, you can create very powerful scripts; for example, ones which include conditional statements. Such statements let you control the execution of programs depending on certain conditions. For details on conditional CLI statements, see Section 2, "Advanced Operations."

Using Edit to Create and Run a Script

Let's assume that you want to create a script that will change the system default pass count value to 2, set the error class to warning, and display the contents of the System Equipment Table, including parameters. In addition, you want to see a message appear that will tell you that these changes have been made.

Name the file TRIAL. To use the Edit utility to create this script, type:

```
ADES-CLI> EDit TRIAL.SRCE ↲
```

The system responds with the question:

```
CREATE NEW FILE? (Y,N) [N]?
```

Type in **Y**, followed by <↲>. You can now write your script. First, you must get into insert mode, where you can type in the appropriate commands. To do this, type:

```
EDIT-CLI> INSERT ↲
```

You can see that you're now in insert mode because the prompt has changed to an asterisk plus a line number as follows:

```
*1 DEFault PAsmax 2 <↲>
*2 EClass Warning <↲>
*3 EQuipment Read <↲>
*4 Write CHANGES ARE INCORPORATED. <↲>
*5 $
```

NOTE: On line five, the user presses the ESC key; the system responds by printing out the dollar sign.

You now get the EDIT-CLI prompt back. To save the new file, and return to ADES-CLI, type **BYE**. To execute this file from EDIT-CLI, you would type:

```
EDIT-CLI> Execute ↲
```

You will be asked to confirm execution. The file is not saved if you execute it from EDIT-CLI.

To execute a file from ADES-CLI, you would simply type in the name of the file.

Writing Scripts

When writing scripts, you can incorporate the **READ** and **WRITE** commands. If you want to insert a pause in a macro to allow the operator to do a certain task, use **READ** with the appropriate message as the argument. **WRITE** (see above example) prints a message in the macro and is used to let the operator know that a particular event has taken place.

For complete details of the Edit utility, see Section 4.

Section 2
Advanced Operations

Chapter 1
Advanced Trouble-Shooting

In Chapter 7 of Section 1, we discussed those techniques most commonly used in trouble-shooting. ADES also provides some more sophisticated techniques. Correctly used, these are very helpful. If, on the other hand, you use them carelessly or without fully understanding them, you may complicate and lengthen the trouble-shooting process.

To control the execution of test programs, ADES CLI lets you pass information to them via a switch register or a string. In addition, ADES provides two debuggers to help you track down problems.

The Switch Register

The switch register is a 32-bit register, used for passing numerical information to and from diagnostic programs. (Do not confuse this register with the Environment Control Word, which defines the operating environment.) To find out how to define the switch register for a particular test program, look at the HELP file for the program.

Any time you wish to see the contents of the switch register, use either the **STATUS** or **SWREG** command. To change the contents, use the **SWREG** command, and an argument.

You have three options for displaying or storing values of the switch register. To display the contents of the 32-bit switch register, you would use the **SWREG** command with no arguments. To store a specific 16-bit value in **SWREG**, you would use the **SWREG** command along with a 16-bit value. To store a 32-bit value in the switch register, use **SWREG** followed by the two appropriate 16-bit values.

The String

The string is a buffer (memory storage area) that can hold up to 80 characters. You can use the string to pass textual information to a program. To find out what the string contains, use the **STATUS** or **STRING** commands. To create a string, use the **STRING** command; to clear it, use the **CLEAR** command.

The Symbolic Debugger

The Symbolic Debugger (ADEB) lets you debug programs by accessing and changing the contents of any internal register or memory location. You can do this by controlling a program's execution with breakpoints (which stop the program at specific points). By placing breakpoints in a program, you can monitor a test program's progress and examine the operation of sub-tests. The Debugger lets you insert up to eight different breakpoints in a program. After a test program halts at a breakpoint, you can examine or change the values in memory or in certain registers. Then, you can restart the test program. Breakpoints defined with ADEB are maintained until you instruct ADEB to remove them.

When you run the Debugger, the CLI loads it first, followed by the ADES test program. If the program is too long to fit, you will not be able to run it with the Debugger. (ADEB requires 2.5 KW. On a machine with 16 KW of memory, you have 5.5 KW available to you; on a machine with 32 KW of memory or more, the edit file can be 21.5 KW.)

For more details, see Section 4 of this manual, or refer to the *Symbolic Debugger User's Manual* (093-000044), or the *ECLIPSE Symbolic Debugger* (093-000140).

The Octal Debugger

The Octal Debugging Tool (ODT) provides some limited debugging facilities. The advantage of using the ODT rather than the Symbolic Debugger is that the ODT does not use any ADES system routines. Because the ODT is part of the resident system code, you may be able to run programs with the ODT that you cannot run with ADEB. The ODT, however, can be executed only from a primary teletype (device code 10/11), when running from a NOVA® or ECLIPSE® type processor. Breakpoints defined with the ODT are automatically removed once the breakpoint is entered.

You can also use the ODT for system restarts and memory dumps, as discussed in Section 4, Chapter 4, "The Octal Debugging Tool (ODT)." For more details on both debuggers, see Section 4.

Chapter 2
Conditional CLI

In Chapter 8 of Section 1, we gave you some examples of scripts and the kinds of commands you use within them. Here we will explain how to create even more powerful scripts through the use of conditional CLI commands. In other words, you can set up scripts so that commands are executed only if a condition is met. The condition is the status of a software control flag.

ADES includes six software control flags: R0, R1, R2, R3, WARNING, and ERROR. These flags can all be accessed by user programs (so the flags can be set from within a program). The ERROR flag is set automatically whenever a diagnostic program finds an error; the other flags can be set with the **SET** command. In addition, the WARNING flag is set whenever the CLI encounters an error in a command line. You can clear any flag with the **CLEAR** command.

To check the status of any flag, use the **TEST** command. Depending on the result of the test, the CLI then continues at a particular node, or point, in the script. You define these nodes with the **LABEL** command.

Figure 2.1 shows you a script that contains conditional CLI statements. The purpose of this script is to run a test of the ECLIPSE® Standard Instruction Set and test for an error. The conditional statements in the script will ensure that if there is an error, the R0 flag is set and a message is printed out. If there is no error, the R0 flag is cleared, and a different message is printed.

```
ADES-CLI> TYpe TEST.SRCE

   WRITE **Testing ECLIPSE Standard Instruction Set**
   X EASIS_X1
   X EASIS_X2
   X EASIS_X3
   X EASIS_X4
   TEST ERROR YES
   CLEAR R0; WRITE **TESTS PASSED**; TEST R0 EXIT EXIT
   LABEL YES; SET R0; WRITE **TESTS FAILED**
   LABEL EXIT
```

Figure 2-1. A Sample Conditional CLI Script

In the above script, the first line prints out on your console. Then, the four ECLIPSE® standard instruction sets are executed sequentially. Next, the ERROR flag is tested to see whether it is set to true or false. If the flag is true, then the R0 flag is set and the message TESTS FAILED is printed out. If it is false, the R0 flag is cleared, and the message TESTS PASSED is printed out. The script ends at EXIT whether this flag is set at true or false.

For further details on the **SET**, **CLEAR**, **TEST**, and **LABEL** commands, see Section 3.

Section 3
ADES CLI

Chapter 1
Using the CLI

The ADES CLI lets you control a number of system variables, such as the system console, the radix used by the operating system, and the way errors are reported by test programs. The CLI also lets you manipulate the many kinds of files that ADES supports. In addition, CLI commands let you execute system utilities (such as the Edit utility or Script Builder), as well as test programs.

You can use the CLI in two modes: manual and script. While in manual mode, the CLI processes commands that you enter in response to the CLI prompt, ADES-CLI>. In script mode, the CLI processes commands that it finds in a macro file that you previously created. Macros provide short-cuts to frequently executed sequences of commands (refer back to Section 1, Chapter 8, "CLI Macros").

Command Formats

You must observe certain rules when entering CLI commands. CLI command lines consist of a command alone, or of a command followed by one or more arguments. An argument is a piece of information the CLI requires to execute the command; for example, the name of a file. Some commands require arguments, others allow them, and still others do not accept them.

When entering CLI commands, you can mix uppercase and lowercase letters; the CLI does not distinguish between them. When entering numerical information, however, remember to use the current radix (octal is the default, but you can change this, as described in Chapter 2 of this section). If you want to change the radix of a particular number, add a radix indicator to the number:

K	octal radix, for example, 7760K
.	decimal radix, for example, 18349.
H	hexadecimal radix, for example, 123A9H

If your command line has one or more arguments, you must separate the commands and the argument(s) with one or more spaces, or a tab, or a comma.

For example,

```
ADES-CLI> ECLASS WARNING
```

or,

```
ADES-CLI> ECLASS,WARNING
```

Terminate (transmit) each command by pressing the Carriage Return (CR), NEW LINE, Escape (ESC), or Line Feed (LF) key. The CLI takes no action until you press one of these keys. Once you do so, the CLI tries to interpret the command.

Abbreviations

You may abbreviate commands and their arguments, if you wish, as long as the CLI cannot confuse them with any other commands or arguments. For example, you can abbreviate the **DELETE** command as **DEL**, but not as **DE**, because that would not distinguish it from the **DEFAULT** command. When we explain how to use each command, we indicate abbreviations in uppercase letters, for example, **DELete**.

Multiple-Command Lines

If you wish to put more than one CLI command on a line, you can do so by separating commands with a semicolon (;). The last command must be terminated by a CR, NEW LINE, ESC, or LF character. Commands are executed in order. For example:

```
ADES-CLI> LOG;MEDIA
```

first displays the status of logging and then displays the current runtime media. Note that if you have a sequence of commands on a line, and one contains an error, the CLI processes only those commands (if any) preceding the one with the error (unless the CLI error class is warning or ignore). Note also that you can fit only 68 characters on one command line.

Screen Edit

On video terminals you can use the Screen Edit utility to alter, delete, or repeat the command line. Most of the commands this utility accepts consist of control characters. To issue any control character, first press and hold the CTRL key, then press the appropriate character key. For example, to issue a CTRL-U, press and hold CTRL and then press U. If you use a command key incorrectly, or if you try to exceed 68 characters, the bell will ring to notify you of your mistake. Table 1.1 summarizes the Screen Edit control characters.

Table 1-1. Screen Edit Commands

Character	Action
CTRL-A	Moves the cursor to the end of the current command lıne, if the command line above it is shorter. If the command line above is longer, CTRL-A repeats the longer part of the command line on the current line. If the cursor is on a blank line, it repeats the entire preceding line.
NOTE: You cannot repeat a command line after a CTRL-C, CTRL-A or CTRL-C, CTRL-B sequence.	
CTRL-B	Positions the cursor at the beginning of the preceding word.
CTRL-E	Starts insert mode. This mode lets you insert characters in the middle of a command line. When you finish inserting, use CTRL-E again to turn off insert mode. In insert mode you can use only CTRL-E, CTRL-J, CTRL-L, CTRL-M, DEL, CTRL-U, and ESCAPE.
CTRL-F	Positions the cursor at the beginning of the next word.
CTRL-H or HOME	Positions the cursor at the beginning of the line.
CTRL-J or NEWLINE	Transmits command. Turns off insert mode (if it was on) and uses the entire line as the command.
CTRL-K or ERASE EOL	Erases all characters to the right of the cursor.
CTRL-L	Same as CTRL-J.
CTRL-M or CR	Same as CTRL-J.
CTRL-U	Deletes the entire line.
CTRL-X or →	Moves the cursor one position to the right.
CTRL-Y or ←	Moves the cursor one position to the left.
ESC	Deletes the entire line, prints $, and issues a new prompt.
DEL or RUBOUT	Deletes the preceding character.

Making Corrections

On a hardcopy terminal, if you make an error while typing a command line, you can correct it with either the DEL or RUBOUT key. When you press one of these keys, an underscore (_) appears, indicating that the preceding character is being deleted. To delete an entire line, use CTRL-U, which produces a ^U and moves the cursor to the beginning of the next line.

Control Characters

In addition to the Screen Edit control characters, there are other control characters which tell the operating system to take some action, such as interrupting the execution of a program. Table 1.2 summarizes these control characters.

Table 1-2. Control Characters (continues)

Character	Action
CTRL-C CTRL-A	Terminates any test program or utility that is currently running. If the CLI is processing a multiple-command line or a script file (see above), the program goes on to the next command. If no command follows the one which was terminated, the CLI issues another prompt. When you issue a CTRL-C CTRL-A, the message ABORT appears.
CTRL-C CTRL-B	Aborts a script. The currently running diagnostic is aborted and all other CLI commands are ignored. The message ABORT appears, followed by a CLI prompt. If no script is running, CTRL-C CTRL-B has the same effect as CTRL-C CTRL-A.
CTRL-C CTRL-C	Prints the characters ^C^C if ADES or a test program is still running. This control character sequence is very useful in checking whether the system is still up or a program has hung.
NOTE: The response of the system may not be immediate.	
CTRL-D <0,...,9>	Complements the corresponding bit (i.e., 0-9) in the 32-bit Switch Register.
CTRL-D <A,...,V>	Complements the corresponding bit (i.e., 10-31) in the 32-bit Switch Register.
NOTE: To denote a character that is to act as a Switch Register function key, the symbol "&" is printed before the character. For example, typing CTRL-D 8 echoes on the system as ^D&8. On video displays, the characters are dimmed or blinking, to distinguish them from non-control characters.	
CTRL-D CTRL-M	Prints the contents of the 32-bit Switch Register in the current radix.
NOTE: CTRL-D is valid in both input mode and polling mode.	

Table 1-2. Control Characters (concluded)

CTRL-O	Disables interrupts if they were on and executes the octal debugger. For details see Section 4, Chapter 4, The Octal Debugging Tool (ODT).
CTRL-P	Interprets the next character as a single keystroke command. For details see Section 1, Chapter 4, "Controlling the Operating Environment." If the next character is a control character (i.e., less than 40 octal), do not process the character to see if it is an ADES control character; instead, input the character literally.
WARNING: CTRL-P followed by either NEW LINE, CR, or formfeed is functionally equivalent to ";" when parsed by the CLI.	
CTRL-Q	Resumes output to the console (undoes the effect of CTRL-S).
CTRL-S	Suspends output to the console (to resume output, use CTRL-Q).

If You Need Help

We remind you that ADES CLI includes a **HELP** command, plus a tutorial mode, should you need some help in using ADES.

You can use the **HELP** command any time the CLI is in manual mode; the command will give you information about CLI commands, related topics, system programs, and test programs.

Secondly, ADES CLI provides tutorial help whenever a system flag called OPERATOR is on and you omit arguments to a command which requires them. The CLI then steps you through a tutorial session so that you understand the arguments you must provide. When you bring ADES up, OPERATOR is automatically on, so you will get tutorial help unless you turn this flag off (see **OPERATOR** command in the next chapter.)

Chapter 2
CLI Commands

This chapter is a complete reference to all CLI commands. We summarize the commands by function, and then provide a detailed description of each command. The commands are in alphabetical order for easy reference.

Getting Help

Command	Format	Meaning
HELP	*[item]*	Displays information about item.

Working with the System Equipment Table

Command	Format	Meaning
EQUIPMENT	*[VERBOSE [device_code]]* *[WRITE [filename]]* *[READ [filename]]* *[FLUSH]*	Displays, saves, replaces, or clears the System Equipment Table (SET).
PROTECT	*device_code [unit_number]*	Protects a device against writing.
SCRATCH	*device_code [unit_number]*	Makes the specified disk or tape a SCRATCH media.
UNPROTECT	*device_code [unit_number]*	Removes write protection from a device.

Controlling the Operating Environment

Command	Format	Meaning
CONSOLE	[device_code [line_number]]	Displays or sets the system console.
	[slave_dev_code mux_dev_code line_number]	
DEFAULT	[<PASMAX,PORT> [new_value]]	Displays or sets a system default.
ECLASS	[ERROR]	Displays or sets the CLI error class.
	[WARNING]	
	[IGNORE]	
ENVIRONMENT	[BRIEF [new_value1][new_value2]]	Displays, sets, or clears one or more flags in the Environment Control Word.
	[INITIAL]	
	[<CLEAR,OFF,NO> flag1 .. flagn]	
	[<SET,ON,YES> flag1 .. flagn]	
LOG	[CLEAR [device_code [unit_number]]]	Displays, enables, or disables console logging; or clears a console log.
	[START [device_code [unit_number]]]	
	[DISPLAY [device_code [unit_number]]]	
	[STOP]	
	[SIZE [new_size]]	
NOTIFY	[<SET,ON,YES> <All, Tests, Modules, Passes>] flag1 .. flagn]	Displays or defines the level of notification.
	[CLEAR ALL] flag1 ..flagn]	
OPERATOR	[<SET,ON,YES>]	Displays or defines the status of the operator flag.
	[<CLEAR,OFF,NO>]	
RADIX	[8]	Displays or defines the radix.
	[10]	
	[16]	
SECONDARY	[device_code [line_number]]	Displays, defines, or undefines the secondary output device.
	[slave_dev_code mux_dev_code [line_number]]	
	[UNDEFINE]	
STATUS		Displays status of software control flags, runtime media, radix, switch register, string, CLI error class.
SYSID	[new_system_id]	Displays or defines the system identification message.

Manipulating Files

Command	Format	Meaning
APPEND	*dest_file source_file*	Appends one script source file to another.
COPY	*dest_file.filetype source_file.filetype*	Copies a file.
DELETE	*[template]*	Deletes one or more files.
FILESTATUS	*[template]*	Lists all or some files on runtime media.
PERMANENCE	*template [ON]* *[OFF]*	Sets or displays a file's permanent status.
RENAME	*current_name.filetype newname*	Changes a file's name.
SPACE		Displays amount of disk space in use, not usable, and free.
TDUMP	*filename.filetype [device_code [unit [file#]]]*	Dumps a file to the tape drive indicated.
TLOAD	*filename.filetype [major_rev[:minor_rev] [device_code[unit[file#]]]*	Loads a file from the specified file number on the drive indicated.
TYPE	*filename.filetype*	Displays a text or script source file.

Working with Media

Command	Format	Meaning
MEDIA	*[device_code [unit_number]]*	Displays or sets the runtime media.
PROTECT	*device_code [unit_number]*	Protects a device against writing.
SCRATCH	*device_code [unit_number]*	Makes the specified disk or tape a SCRATCH media.
STATUS		Displays status of software control flags, runtime media, radix, switch register, string, CLI error class.
UNPROTECT	*device_code [unit_number]*	Removes write protection from a device.

Bringing Up ADES, Executing Programs, and Shutting Down ADES

Command	Format	Meaning
ACCEPT*		Runs acceptance tests (one pass per program).
BOOT	*[device_code [unit_number]]*	Shuts down system and boots a device. To boot AOS, you must also set hard console switches (or switch register on soft console) to device code of media you're booting.
BYE	*[NOHALT]*	Shuts down system and optionally restarts CLI.
CHECKSUM	*[pathname]*	Checks the integrity of the resident system code or specified file.
DEFAULT	*[<PASMAX,PORT> [new_value]]*	Displays or sets a system default.
ERMES	*error_code*	Prints explanation of error.
LOAD	*filename [address]*	Loads a file into memory.
RUN*		Runs standard set of test programs (three passes per program).
SLAVE	*device_code filename [address]*	Loads file into slave processor's memory and optionally starts slave processor.
START	*address*	Starts the loaded program.
[XEQ]	*filename [passes [address]]*	Executes a program.

*these are really macros, not commands.

Writing Scripts

Command	Format	Meaning
CHAIN	*script_filename[.srce] [macro_arguments]* **	Loads in and executes the specified script file.
CLEAR	*STRING* *ALL* *flag1 .. flagn*	Clears one or more control flags or the string.
CMPSTR	*<R0,R1,R2,R3,WARNING> string*	Compares the string on the command line with the current string buffer.
COMMENT	*comments*	Enters a comment.
LABEL	*label_name* **	Names a node in a conditional script.
RDSCF	*<R0,R1,R2,R3,WARNING> [message]*	Prints the optional message and inputs the new value for the specified software control flag.
RDSWR	*[message]*	Prints the optional message and inputs a string containing one or two 16-bit values to store into SWREG.
READ	*[message]*	Prints an optional message, takes input from the operator, and stores it in the string.
SET	*argument_1 [<+,-,=,&>,argument_2]*	Sets some or all of the software control flags (optionally to value of expression).
STRING	*[new_string]*	Sets or displays the string.
TEST	*flag true_label [false_label]* **	Tests a software proceeds to a node in the script.
WRITE	*[message]*	Prints a blank line or specified message.

**Valid only within a script.

Communicating with Programs

Command	Format	Meaning
STRING	*[new_string]*	Sets or displays the string.
SUBINFO	*filename[.prg]*	Prints the submittal information for the specified ADES program.
SWREG	*[16_bit_value [16_bit_value]]*	Sets or displays the switch register.

ACCEPT

Description

A macro that executes diagnostic programs on all equipment listed in SET. Be sure that the disk or tape drives being tested are not write-protected.

NOTE: Some equipment cannot be tested in auto-sequence script because of program restrictions.

APPEND

Syntax

APpend *dest_filename[.SRCE] source_filename[.SRCE]*

dest_filename is the name of the source file to which you are appending another. The *source_filename* is the name of the source file you are appending. You can optionally include the .SRCE extension.

Description

Appends one source file to another.

Example

ADES-CLI> **AP** *RICH.SRCE ANNA*

Appends a source file called *ANNA* to one called *RICH.*

BOOT

Syntax

BOot *[device_code [unit_number]]*

device_code is the device you wish to boot. If you do not provide a device code, the CLI assumes you want to bootstrap the runtime media. The *unit_number* is the number of the device; if omitted, unit 0 is assumed.

Description

Shuts down the operating system in an orderly fashion and boots the specified device.

CAUTION: In order to boot AOS from ADES, you must set the hard console switches (or the switch register in a soft console) to the device code of the AOS media you're booting.

Examples

ADES-CLI> **BOOT** *27 1*

Boots unit 1 on device code 27.

ADES-CLI> **BOOT**

Boots the system media.

BYE

Syntax

BYe *[Nohalt]*

Selecting the *Nohalt* option allows you to get back to ADES CLI by pressing NEW LINE. (If *Nohalt* is not selected, the processor is halted.)

Description

Shuts down the system. The console log is closed; the runtime media is rewound/recalibrated.

Example

ADES-CLI> **BYe**

You cannot continue execution.

CHAIN

Syntax

CHAin *script_filename[.srce] [macro_arguments]*

Description

Loads and executes the specified CLI source file. If the file type extension is excluded, file type .SRCE is assumed. **CHAIN** is used in a macro to explicitly direct the sequencing of macros. (This command in valid only in Script Mode.)

Macro_arguments are passed to the specified macro.

Example

```
ADES-CLI>              TYPE MACRO.SRCE
                       WRITE HELLO
                       MACROA
                       WRITE GOODBYE.

ADES-CLI>              TYPE MACROA.SRCE
                       WRITE HOW ARE YOU
                       CHAIN MACROB
                       WRITE THIS DOES NOT PRINT

ADES-CLI>              TYPE MACROB.SRCE
                       WRITE FINE

ADES-CLI>              MACROB

HELLO
HOW ARE YOU
FINE
GOODBYE.
```

CHECKSUM

Syntax

CHEcksum *[pathname]*

Description

Verifies the integrity of the memory resident operating system code by summing all the memory resident code and comparing the total to a known value. When used with the optional argument *pathname*, **CHECKSUM** performs a checksum on the specified file. If a FILE CHECKSUM ERROR is returned, then the checksum word for the file will have been updated to reflect the new contents of the file.

WARNING: This command only catches single bit errors.

Example

ADES-CLI> **CHE**

Performs an immediate checksum on the resident operating system.

CLEAR

Syntax

CLear *[String]*
[All]
<R0, R1, R2, R3, Warning, Error>

Description

A conditional CLI command. Depending on the argument, **CLEAR** clears the string buffer (i.e., sets the current string to null), clears the specified software control flags (i.e., R0, R1, R2, R3, Warning, and/or Error), or clears all the software control flags.

Example

ADES-CLI> **CL** *R2 ERROR*

The *R2* and *ERROR* flags are cleared.

CMPSTR

Syntax

CMpstr *<R0,R1,R2,R3,WARNING> [string]*

Description

CMPSTR compares the *string* on the command line with the current string buffer. If the two strings match, the specified software control flag is set to true (= +1); otherwise, the software control flag is set to false (=0).

If *string* is not supplied, **CMPSTR** sets the specified software control flag to true if the first byte of the current string buffer is null; otherwise, the software control flag is set to false.

CMPSTR can be executed only from a script. If you execute the command in manual mode, ADES returns a COMMAND NOT VALID IN MANUAL MODE warning.

If no arguments are supplied, ADES returns a COMMAND REQUIRES ARGUMENTS warning.

Example

ADES-CLI> **CMPSTR** *R1 ERCC ERROR*

Compares *ERCC ERROR* with the current string buffer.

COMMENT

Syntax

COMment [*anything useful*]

Description

When writing scripts, **COMMENT** allows you to insert the comment of your choice into a CLI macro. If console logging is enabled, **COMMENT** lets you enter comments into the console log.

Example

ADES-CLI> **COM** *Problem occured on 1-10-82.*

Assuming console logging is enabled, the comment *Problem occurred on 1-10-82.* is entered into the console log.

CONSOLE

Syntax

CONsole *[slave_device_code [device_code [line_#]]]*

Description

Used alone, **CONSOLE** displays the current system console type (VIDEO, or HARDCOPY), model number, device code (input and output if the device is a TTI/TTO), and, if the device is a MUX, the line number and its characteristics. (Console type, model number, and line characteristics information is derived from the SET.) If the system console is on a slave processor bus, the device code of the slave processor is printed in parenthesis after the multiplexor's device code.

With the argument *device_code*, **CONSOLE** changes the system console to the device code specified. The console type is set according to the SET. The device code specified refers to the input device (e.g., CONSOLE 10).

With the arguments *device_code line_#*, **CONSOLE** changes the system console to the device code and line number specified.

With the arguments *slave_device_code device_code line_#*, **CONSOLE** changes the system console to the line, device code, and slave device code specified.

NOTE: The device codes are parsed according to the current radix. The line number is always parsed in decimal.

Example

ADES-CLI> **CON** *65 34 1*

Switches the console to the terminal on line 1, MUX 34, and IOP device code 65.

COPY

Syntax

COPy *destination_pathname srce_pathname*

The original file is *srce_pathname* and the new file is *destination_pathname*. Copies *srce* to destination.

Description

Copies the original file into a new file. The new file must not already exist. The file type of the new file does not have to match that of the original file.

Example

ADES-CLI> **COP** *book.txt preface.togl*

Copies the file preface to the file book.

DEBUG

Syntax

DEBug *filename[.prg] [start_address]*

Description

First loads the ADES Debugger and then the ADES program, starting at location 0. Next, transfers control to the Debugger.

NOTE: A warning is issued if there is not enough memory to hold both the ADES Debugger and program, or if you provide an illegal filetype extension (it must be .PRG).

If you provide only a filename, this command loads the Debugger at the top of user space, loads the ADES program at location 0, prints the start address of the Debugger, and starts the Debugger.

If you add the *start_address* argument, the command loads the Debugger at the address specified, then loads the ADES program at location 0, prints the start address of the Debugger, and starts the Debugger.

Example

ADES-CLI> **DEB** *INT.PRG*

Loads the Debugger at top of user space, loads program INT at location 0, prints start address of the Debugger, and starts it.

DEFAULT

Syntax

DEFault *[<PASMAX,PORT> [new_value]]*

Description

Displays or sets system default values. Alone, **DEFAULT** displays the current system default values: *PASMAX* or *PORT*. *PASMAX* is the number of passes a test program makes when executed via an **XEQ** *filename* command. A *PASMAX* of -1 implies "loop forever." *PORT* is the default user port (I/O channel). The maximum valid port number is 6.

If you use **DEFAULT** with either *PASMAX* or *PORT*, it displays the specified default value. If you provide a *new_value*, **DEFAULT** sets the specified default value to *new_value*.

Example

ADES-CLI> **DEF** *PASMAX 2*

Sets the default value at 2; programs make 2 passes if the pass count on the **XEQ** command is omitted.

DELETE

Syntax

DELete *template[.filetype]*

Description

Eliminates file(s) specified from runtime media. You can use templates to indicate which files to delete. The system will require you to confirm if you're in manual mode; if the command is executed from a macro, you do not need to confirm.

Example

```
ADES-CLI> DEL +.srce
```

Deletes all source files from runtime media.

ECLASS

Syntax

EClass *[Ignore, Warning, Error]*

Description

Sets or displays the current CLI error class, which determines whether or not a script or a long command line will be continued if an error is detected. The error class is initially set to ERROR.

If the error class is IGNORE, the script or command line will continue despite the error and you will not see any error message. If the CLI error class is set to WARNING, the script or command line will, again, be continued, but this time a WARNING message is printed. Setting the class to ERROR will cause the script or command line to be aborted and an ERROR message is printed.

Example

```
ADES-CLI> EC
```

Displays the current error class. (You can also use the **STATUS** command to display the error class.)

EDIT

Syntax

EDit *filename.filetype*

Description

Edits the specified file. Data in source or text files are represented as ASCII characters; in all other file types, as numbers according to the current radix.

A warning is given if:

- The file does not exist and ADES couldn't create a new one. The message, FILE CREATION ERROR is printed, along with an explanation. Execution continues in EDIT-CLI, but the edits cannot be saved.
- The operator flag is off.
- The **EDIT** command was used in a macro.

If you do not provide a filetype extension, the question FILE TYPE? will appear. You can answer with any one of the fourteen file types. You have the choice of creating a new file if you specify a SRCE, TXT, or TOGL file and there is enough space on the runtime media and in the directory.

WARNING: If a file checksum error is encountered while reading in the edit file, then the session is aborted if the CLI error class is ERROR. If the error class is WARNING, the system prints a warning message but continues to execute the edit utility.

Example

ADES-CLI> **ED** *ART.TXT*

Allows you to edit the text file, *ART*.

ENVIRONMENT

Syntax

ENvironment *[Brief [new_value1][new_value2]]*
[Initial]
[<Clear, Off, No> [flags]]
[<Set, On, Yes> [flags]]

flags is a list of one or more Environment Control Word (ECW) flags.

Description

Allows you to modify and/or display the current operating environment, which is controlled by flags contained in the Environment Control Word (ECW). This command lets you define the value in the following flags: LOOP, CONSOLE, PERCENT, SECONDARY, PAUSE, REPORTALL, TERMINATE, VERIFY, PAGE_MODE, ABORT and DCHANNEL. See Section 1, Chapter 4, "Controlling the Operating Environment," for more details.

Without an argument, **ENVIRONMENT** prints the current operating environment, including the appropriate single keystroke command and the value of the ECW.

With the argument *Brief*, it prints the operating environment, but only includes the value of the ECW; the contents of the ECW are set according to *new_value. new_value1* sets the contents of bits 0-15 in the ECW. If you add *new_value2*, bits 16-31 in the ECW are set as well. In either case, the value of bit 10 (console logging) is not affected.

With the argument *Initial*, it restores all flags to their original value. (See Section 1, Chapter 4, "Controlling the Operating Environment," for a table containing the initial value of the flags.) You can clear the indicated flags with the arguments *Clear, Off,* or *No*, and set the indicated flags with the arguments *Set, On,* or *Yes*.

NOTE: In addition to using **ENVIRONMENT** to define the flag in the Environment Control Word, you can change the value of a flag by using single or ouble keystroke commands. Refer to Section 1, Chapter 4, "Controlling the Operating Environment" for details.

Example

ADES-CLI> **EN** *CLEAR SECONDARY LOOP*

Clears the secondary and loop flags.

EQUIPMENT

Syntax

EQuipment *[Verbose [device_code]]*
[Write [filename[.udf]]]
[Read [filename[.udf]]]
[Flush]

Description

Allows you to save, display, write over, or clear the contents of the SET. Without arguments, this command displays the contents of the SET, excluding device dependent information.

Verbose adds the display of device dependent information.

device_code lets you display the information for whatever device you specify. For devices without a code, such as -- CHAR, use device code 77 (octal).

Write is used to update the file EQUIP0, which contains the contents of the current SET. This file is created if it does not already exist. If you add a filename, the contents are written to the file specified. This must be a UDF type file; it is created if it doesn't exist yet.

Read replaces the current SET with the contents of EQUIP0. Adding a filename to **EQ** *Read* replaces the SET with the contents of the file you specify. Again, the file type must be a UDF.

Flush clears the contents of the SET so that a new SET can be generated manually.

Example

ADES-CLI> **EQ** *W*

Writes contents of the SET into the file EQUIP0; creates the file if it doesn't exist. (See Section 1, Chapter 3, "The System Equipment Table" for more information.)

ERMES

Syntax

ERmes *error_code*

Description

Prints an explanation of the specified error code.

Example

ADES-CLI> **ER** *23*

Explains error code 23.

FILESTATUS

Syntax

Filestatus *[template [.filetype]]*

Description

Lists the file type, file name, revision numbers (major/minor), and size (number of 256-word pages) of the specified files on the runtime media. You can provide a template character (a * or a +) as well as a file name or a file type to specify certain files you want to see listed. Section 1, Chapter 5, "The File System" explains template characters more fully.

Examples

ADES-CLI> **F** +*.srce*

Lists all source files on the runtime media.

ADES-CLI> **F** *tes**

Lists all files beginning with *tes* plus one other character.

HELP

Syntax

Help *[item]*

Description

Provides information on a specified item (topics, CLI commands, and programs). If you do not provide an argument, a help file is displayed that provides you with information on general operating procedures and how to get further help. Help files for programs include the part number, a description, and the definition of any switch register bits the program uses.

Example

ADES-CLI> **H** *OPERATOR*

Gives you information on the **OPERATOR** command.

LABEL

Syntax

LAbel *label_name*

The *label_name* is 1 to 5 contiguous alphanumeric characters, including all punctuation characters, except ;.

Description

A conditional CLI command used only in scripts. When used with the **TEST** command, **LABEL** marks the point where the CLI will continue to execute commands. The commands following the **LABEL** command are executed if the control flag last executed transferred control to the label name. If no control flag is currently being tested, the commands are also executed. **LABEL** may not be used in manual mode.

Example

```
ADES-CLI>      TYPE LABEL_EX.SRCE
               SET R0
               TEST RO TRUE
               WR THIS IS NOT PRINTED.
               TEST R0 NEXT NEXT
               LABEL TRUE
               WR RO IS TRUE.
               LABEL NEXT

ADES-CLI>
LABEL_EX RO IS TRUE.
ADES-CLI>
```

LOAD

Syntax

LOad *filename[.filetype [address]]*

Description

Loads the file from the runtime media into memory at the address given, or into location 0 if no address is given. You will get an error if the file is too big, the file does not exist, or if there was an error when the file was read in from the runtime media. If you do not provide a file type, ADES looks under file type .PRG only. Here are some warnings regarding this command:

- The **LOAD-START** method of executing is not recommended (use **XEQ**)
- Do not use **HELP** after you have loaded a file with **LOAD**; **HELP** will write over the file you've loaded.

Example

ADES-CLI> **LO** *NANC.PRG*

Loads a file called *NANC* into memory at location 0.

LOG

Syntax

LOG *[STOp]*
[Clear [device_code] [unit]]
[STArt [device_code] [unit]]
[Display [device_code] [unit]]
[Size [new_size]]

Description

Lets you save all output sent to the system console or the secondary output device. Console logging is enabled when output is being copied to the Console Log; it is disabled when output is not being copied. (When ADES is first brought up, console logging is disabled.)

On disk media, the Console Log is saved in a disk file called CONLOG (file type RSYS). Alternatively, the Console Log can be written out to a log tape; the runtime media in this case can either be tape or disk.

LOG, without arguments, prints the current status of console logging, and if enabled, to what device code and unit number.

With the argument *Clear*, **LOG** clears all entries from the runtime media console log file. *Clear device_code* clears all entries from the log tape on the specified device. If you add the unit argument, all entries are cleared from the log tape on the specified device and unit number; if the unit is not specified, 0 is assumed.

With the argument *Start*, **LOG** starts or continues console logging to the disk file CONLOG. A console log entry is made to record that console logging was started. This sets the ENABLE CONSOLE LOGGING flag in the Environment Control Word. *Start device_code* starts or continues console logging to the tape drive located on the specified device code, unit 0. *Start device_code unit* starts or continues console logging to the tape drive located on the specified device code, and the specified unit.

LOG *stop* disables console logging and clears the flag in the Environment Control Word. A console log entry records that console logging was stopped. A warning is issued if logging was already disabled.

LOG *display* prints the disk console log stored in CONLOG. The runtime media must be a disk; youll get a warning if its not. The console log cannot be displayed if console logging is enabled. With *display device_code*, **LOG** prints the console log found on the specified tape drive, unit 0. With *display device_code unit*, **LOG** prints the console log found on the specified tape drive, and the specified unit.

LOG *size* displays the current size of the console log file (CONLOG.RSYS). This command is only valid when the runtime media is a disk.

LOG *size new_size* changes the size of the disk console log file to the size specified. *New_size* represents the number of sectors to make CONLOG.RSYS, and is interpreted in decimal (unless a radix indicator is used). Changing the console log size has the following requirements and effects:

1. The new size must be between 8 and 32767; otherwise, an INVALID ARGUMENTS error is returned, and no action is taken.
2. The new size must be large enough to accommodate all the currently used sectors in the console log plus one; otherwise, an ILLEGAL FILE SIZE error is returned and no action is taken.
3. If the new size satisfies requirements 1 and 2, and a CONSOLE LOG OVERFLOW condition was present, then console logging is restartable via a **LOG** *start* command (without having to perform a **LOG** *clear*) when the **LOG** *size new_size* command is finished.
4. The console log size should be returned to 8 sectors before doing any media builds. If not, CONLOG.RSYS won't get transferred to the build media because it's a contiguous file which exceeds 2 KW. If you get an ILLEGAL FILE SIZE error when you try to reduce the log size to 8 sectors, you must first clear the console log, so that it is large enough to accommodate all the currently used sectors plus one.

Example

ADES-CLI> **LOG** *Start 22*

Starts or continues console logging to the tape on device code 22, unit 0.

MEDIA

Syntax

Media *[device code [unit number]]*

Description

Without an argument, **MEDIA** displays the runtime media. The display includes the model number, device code, and unit number.

With an argument, **MEDIA** lets you set the runtime media to the specified device code and/or unit number. Console logging is disabled by the **MEDIA** command (when used with an argument). The media device specified must be listed in the SET as ADES SYSTEM MEDIA. If it is not, a warning is issued. ADES must be able to read the media's Primary Label Block; if it can't, a warning is issued and the original media is maintained.

Example

ADES-CLI> **ME** *33*

Changes the runtime media to device code 33, unit 0.

NOTIFY

Syntax

Notify *[<Clear,Off,No> [All]]*
<Modules, Passes>]
[<Set,On,Yes> [All]]
<Tests, Modules, Passes>]

Description

Displays or defines the level of notification. The operator can be notified of the start of tests or modules, and/or the end of passes.

Without an argument, **NOTIFY** displays whether the operator is being notified of any of the above.

Set All, On All, or *Yes All* sets the notification level to the start of tests, the start of modules, and the end of passes. *Clear All, Off All*, or *No All* clears the notification level so that no messages are printed. You can *Set* the value of the flag in the Environment Control Word by specifying which level you want set (*Tests*, *Modules*, or *Passes*); you can also *Clear* the specified flag in the ECW.

NOTE: You have the option of using single or double keystrokes to complement the value of the following flags in the Environment Control Word: PASSES, TESTS, and MODULES. Refer to Section 1, Chapter 4, "Controlling the Operating Environment" for details.

Example

ADES-CLI> **N** *Set Tests*

Sets the flag in the ECW to YES; you will be notified at the start of tests.

OPERATOR

Syntax

OPerator *[<ON, SET, YES>]*
[<OFF, CLEAR, NO>]

Description

Displays or sets the status of the OPERATOR flag in the Environment Control Word. If the flag is on, programs prompt you for input; otherwise they do not. In the CLI, you have the advantage of being in tutorial mode when the flag is on. As a result, if you forget to provide arguments to a command which requires them, the CLI takes you through a brief tutorial session.

Without an argument, the CLI indicates whether OPERATOR is on or off; with an argument, the CLI sets OPERATOR to the state you request.

NOTE: You have the option of avoiding the **OPERATOR** command if you use keystroke commands. For details, refer to Section 1, Chapter 4, "Controlling the Operating Environment."

Example

ADES-CLI> **OP** *ON*

Turns the OPERATOR flag on.

PERMANENCE

Syntax

PErmanance *template[.filetype] [ON, OFf]*
filename[.filetype] [ON, OFf]

Description

Displays or sets the permanence of the specified file(s). (Permanent files cannot be deleted; non-permanent ones can.)

To display the permanence of a file, use the command along with the appropriate filename, filetype, and/or template. To set permanence on one or more files, add the *ON* argument. To disable permanence on one or more files, add the *OFf* argument.

Example

ADES-CLI> **PE** *FERG.TXT ON*

Sets the permanence of the text file, *FERG*, to *ON*.

PROTECT

Syntax

PRotect *device_code [unit_number]*

Description

Enables write protection on the specified disk or tape on the device code specified. Unit 0 is assumed if no other number is named. If you issue the command in manual mode, ADES requests a confirmation. ADES verifies that the device specified is write-protected.

Example

ADES-CLI> **PR** *33*

Assuming you issued this command in a script, ADES makes device 33 write-protected and prints the message: DEVICE 33,0 IS NOW WRITE PROTECTED.

RADIX

Syntax

RAdix *[8]*
[10]
[16]

Description

Displays or sets the numbering system used for all numeric input and output. Certain values can not be changed: TEST, MODULE, PASS, and line numbers are always decimal; device codes are always octal. The radix displayed or entered is in decimal. The default radix is octal.

Example

ADES-CLI> **RA** *10*

Sets the numbering system to decimal.

RDSCF

Syntax

RDSCf *<R0,R1,R2,R3,WARNING> [message]*

Description

Prints the optional message and inputs the new value for the specified software control flag. A warning is issued if the OPERATOR flag is off.

Example

ADES-CLI> **RDSCF** *R1 New R1 value?*

Prints the message *New R1 value?* and inputs the new value for R1.

RDSWR

Syntax

RDSWr *[message]*

Description

Prints the optional *message* and inputs a string containing one or two 16-bit values to store into SWREG. If one value is entered it is stored in SWREG, bits 0-15. If two values are entered, the second value is stored in SWREG, bits 16-31.

A warning is issued if the OPERATOR flag is off.

Example

ADES-CLI> **RSDWr** *New SWREG value?*

Prints the message *New SWREG value?* and waits for the operator to input SWREG values and press NEW LINE.

READ

Syntax

READd *[message]*

Description

Prints message and inputs string (up to 80 characters long) from the operator. The OPERATOR flag must be on or a warning is issued. This command is useful in macros when you want to pause to wait for the operator to do something.

Example

ADES-CLI> **COMMENT** *Example assumes device code 22,0 is listed as SCRATCH in SET.*
OPERATOR ON
COMMENT Command not necessary if OPERATOR flag is already on.
READ Hit NEW LINE when scratch tape is mounted on mag tape unit 0.
XEQ MTA0_DUMP

This example shows you a portion of a script file; as you see, **READ** puts a pause in the macro so the operator can, in this case, mount the scratch tape.

RENAME

Syntax

REName *current_filename.filetype new_filename[.filetype]*

Description

Changes the name of a file; the new file name cannot already exist. You must specify the type of the current file. If you specify a filetype for *new_filename*, it must match the filetype specified for *current_filename*.

Example

ADES-CLI> **REN** *greg.txt hughes*

Renames the current text file *greg* to the the new text file *hughes*.

RUN

Description

A macro that tests each device, option, and function of the current system configuration; runs each program for 3 passes.

NOTE: Some equipment cannot be tested in auto-sequence script because of program restrictions.

SCRATCH

Syntax

SCratch *device_code [unit_number]*

Description

Makes the specified disk or tape a SCRATCH media. Unit 0 is assumed if none is specified. If the command is executed in manual mode, a confirmation is requested by ADES before the device is made a SCRATCH media.

After write-enabling the device, ADES verifies the new device status with the message `Device` *dc,un* `is now a SCRATCH media.`

Example

ADES-CLI> **SCRATCH** *22,1*

Makes device code 22, unit 1, a SCRATCH media.

SECONDARY

Syntax

SECondary *[slave_device_code [device_code [line_#]]]*
[Undefine]

Description

Defines, undefines, or displays the secondary output device. When you define a new device code for the secondary output device, it must not be the same as the current system console output device code.

Alone, **SECONDARY** displays the secondary output device.

With the argument *device_code*, **SECONDARY** defines the specified device code to be the secondary output device. Add *line_#*, and the command defines the specified line number of the specified device code to be the secondary output device.

With all three arguments (*slave_device_code, device_code,* and *line_#*), the command changes the secondary output device to the line, device code, and slave device code specified.

With the argument *Undefine* **SECONDARY** undefines the secondary output device.

Example

ADES-CLI> **SEC** *Undefine*
ADES-CLI> **CONSOLE** *sodc*
ADES-CLI> **SECONDARY** *condc*

The system console was on device code *condc*; the secondary device code was *sodc*. The two have now switched device codes.

WARNING: To start printing on the secondary output device, you must set the **SECONDARY** flag in the Environment Control Word, using the **ENVIRONMENT** command. Refer to the entry for **ENVIRONMENT** in this section.

SET

Syntax

SET *argument_1 [[+,-,=,&] argument_2]*

argument_1 is defined as *[All, <R0,R1,R2,R3,Warning>]* and *argument_2* is defined as *[[R0,R1,R2,R3,Warning], Number]*

Description

SET *ALL* sets R0, R1, R2, R3, and the Warning software control flags, to true (which is defined as nonzero); the actual value is +1. If you use **SET** with any one of the flag names as an *argument*, it sets just that flag to +1.

By providing an *argument_1 + argument_2*, you add the value of *argument_2* to the software control flags in *argument_1*. The same applies to *argument_1 - argument_2*, only this time you subtract the values. With *argument_1 = argument_2*, you set the software control flags listed in *argument_1* to the value listed in *argument_2*. For each of these options, if *argument_2* is another software control flag, then the value of that flag is added, subtracted, or equalled.

By providing an *argument_1 & argument_2*, you set the software control flag(s) in *argument_1* equal to the logical AND of the value of the software control flag with the value in *argument_2*. If *argument_2* is another software control flag, then the value of that flag is ANDed with the flag(s) listed in *argument_1*. If *argument_2* is a number, then it is ANDed with the flag(s) listed in *argument_1*.

NOTE: You cannot manually set the Error software control flag.

Example

ADES-CLI> **SET** *All + R0*

Adds the value of R0 to all software control flags.

ADES-CLI> **SET** *R1 R3 - 5*

Subtracts 5 from the software control flags R1 and R3.

ADES-CLI> **SET** *R0 + R0*

Multiplies R0 by 2.

SLAVE

Syntax

SLave *device_code filename[.filetype] [start_address]*

Description

Loads the specified file into the slave processor on the device code given. The device is started at address supplied; it's not started if an address is not given. If you do not supply a file type, ADES looks under type .PRG only. A slave processor that is supported by ADES is IOP, model number 8660.

If the file loaded into the slave processor is an ADES program, the **SLAVE** command also loads the appropriate support routines into the processor and handles all system console input/output to and from the slave processor until the program completes.

Example

ADES-CLI> **SLAVE** *65 IOP_MON.PRG 200*

Loads *IOP_MON.PRG* into the IOP on device code 65 and starts the IOP at location 200.

SPACE

Syntax

SPace

Description

Lists, in decimal, the number of disk sectors currently free, unusable, and already used by files.

Example

ADES-CLI> **SP**

Results in the following:

```
--- CURRENT DISK SPACE (SECTORS) ---
        FREE= xxxxx
        UNUSABLE= xxxxx
        USED= xxxxx
```

START

Syntax

STArt *address*

Description

Starts program execution at the specified address. Used in conjunction with the **LOAD** command.

Example

ADES-CLI> **LOAD** *AMB.PRG; LOAD* **ADEB.OLF** *16000;* **START** *16000*

STATUS

Syntax

STATus

Description

Prints information about the following variables: the software control flags (R0-R3, ERROR, and WARNING), the runtime media, the current radix, the current switch register (32 bits), the current string, and the current CLI error class.

STRING

Syntax

STRing *[new_string]*

Description

Sets or displays the current string buffer (holds up to 80 characters).

Example

ADES-CLI> **STRING** *DATA_FILE.TXT;* **XEQ** *TEST_PROG*

Inserts the string *DATA_FILE.TXT* into the string buffer for use by the program *TEST_PROG.*

SUBINFO

Syntax

SUbinfo *filename[.prg]*

Description

Prints the submittal information for the specified ADES program. The submittal information consists of the program name, part number, major revision level, minor revision level, and submittal.

All submittal information (except the minor revision level) is obtained by examining the Submittal Information Block that resides in locations 202 to 217 (octal) in the specified ADES program.

Example

```
ADES-CLI> SUBINFO PRIOR_X

Filename:       PRIOR_X
Part Number:    096-002500
Revision:       6.2
Submittal:      3

ADES-CLI>
```

The operator has requested submittal information on *PRIOR_X.PRG*. ADES confirms the existence of PRIOR_X.PRG and gives the part number for its listing, 096-002500. The revision level is 6.2, which means that the major revision level is 6, and that Patch #1 and Patch #2 have been installed. (A minor revision level of 0 would mean that no patches have been installed.) The submittal is 3, which means that it is the fourth (i.e., 0-3) submittal of PRIOR_X.

SWREG

Syntax

SWReg *[16_bit_value [16_bit_value]]*

Description

Sets or displays a 32-bit switch register which is accessible to all diagnostic programs. The register is used to pass numerical information to a diagnostic program. (Do not confuse this with the Environment Control Word which controls the operating environment.)

Without an argument, the current contents of the switch register are displayed. To store a specified value in the switch register, supply the *16_bit_value.* To store two separate values in SWREG, supply both values.

Example

ADES-CLI> **SWR** *16_bit_value*

Places the *16_bit_value* in bits 0-15 of the switch register.

SYSID

Syntax

SYsid *[new_system_id]*

Description

Displays or sets the system identification message. You can use the message to note the model number, build date and time, system identification number, and owner.

Example

ADES-CLI> **SYSID** *ECLIPSE S/120 ADES DIAGNOSTIC MEDIA;* **COMMENT** *NEXT COMMAND.*

Changes the system identification to reflect the contents of the ADES system disk. Note that the **COMMENT** command (in addition to the command terminating character ";") does not become part of the new system identification.

TDUMP

Syntax

TDump *filename.filetype [device_code [unit [file#]]]*

Description

Dumps *filename.filetype* to the magnetic tape drive specified. The default values are: device code 22, unit 0, and file 0. The tape drive specified must be listed in the System Equipment Table as SCRATCH.

After the file is dumped, the file name and file type is printed as a verification that the dump completed successfully.

To load the file from the dump tape to an AOS file, use the **COPY** command, specifying an input magnetic tape record size of 8192 bytes (e.g., **COPY/IMTRSIZE=8192 MYFILE @MTA0:0**).

Example

ADES-CLI> **TDUMP** *DATA.UDF 22,0,3*

Dumps the file *DATA.UDF* to the file 3, unit 0, of device 22.

TEST

Syntax

TESt *[R0,R1,R2,R3,Warning,Error] true_label [false_label]*

Description

Used in a script to test the value of one of the six software control flags. If the value of the flag is true, then the CLI continues executing commands after the specified *true_label.* If the value is false, CLI execution contines after the *false_label.* If the flag is false, but no label is provided, the CLI will continue executing at the next statement.

Example

```
WRITE    **TESTING ECLIPSE STANDARD INSTRUCTION SET**
         X EASIS_X1
         X EASIS_X2
         X EASIS_X3
         X EASIS X4
         TEST ERROR YES
         CLEAR RO; WRITE **TESTS PASSED**; TEST RO EXIT EXIT
         LABEL YES; SET RO WRITE **TESTS FAILED**
         LABEL EXIT
```

The **TEST** command is used here to test the error flag. If the flag is set to true, execution continues at *LABEL YES*. If it is set to false, execution continues at the next line.

TLOAD

Syntax

TLoad *filename.filetype [major_rev[:minor_rev] [device_code [unit [file#]]]]*

Description

Loads the file from the specified file number on the magnetic tape drive indicated. The default values are: major revision level 0, minor revision level 0, device code 22, unit 0, and file 0. The tape drive specified must be listed in the System Equipment Table as SCRATCH, WRITE PROTECTED, or UPDATE.

The file will not be loaded onto the runtime media if a file of the same pathname already exists on the runtime media as a permanent file (a CANNOT DELETE PERMANENT FILE error is returned).

The load file on the tape must be in AOS format. Under AOS, use the **COPY** command to dump the file to the tape (e.g., **COPY** @MTA0:0 *filename*). After the file is loaded, the file name and file type is printed as a verification that the load completed successfully.

Files loaded by **TLOAD** are created as non-permanent files (regardless of whether the file existed prior to the **TLOAD**.)

If a device status error is detected by **TLOAD**, the device code, unit number, and DIA status register of the tape drive is printed.

WARNING: TLOAD is a simpler alternative to the Update Utility. However, if you prefer the safety features that the Update Utility contains, do not use the TLOAD command.

Example

ADES-CLI> **TLOAD** *PRIOR_X.PRG 6:1 22,1,2*

Loads the file *PRIOR_X.PRG* (with a major revison level of 6, and a minor revision level of 1) from file 2, unit 1, of device 22.

TYPE

Syntax

TYpe *filename[.srce,.txt]*

Description

Type out the specified source or text file.

Example

ADES-CLI> **TY** *WEASEL.TXT*

Types out the text file *WEASEL*.

UNPROTECT

Syntax

Unprotect *device_code [unit_number]*

Description

Change the status of the specified device code and/or unit number to write-enabled. Unit 0 is assumed if no other unit number is given. If the specified device is hardware write-protected, a warning is issued.

ADES requests a confirmation if command is executed in manual mode. ADES verifies the new status after the device is write-enabled.

WARNING: Be sure to verify that the media you are unprotecting does not contain any DGC or customer software on it. Once the media is write-enabled, the data on the media is vulnerable.

Example

ADES-CLI> **U** *22*

Changes status of device code 22, unit 0, to write-enabled.

WRITE

Syntax

Write *[message]*

Description

Used in macros to print a message, followed by a carriage return. Helpful in informing the operator of the occurrence of an event. A blank line is printed if no message is stated.

If a character sequence with the format *xx* is encountered in *message*, *xx* is treated as a number, and only the ASCII character with the value of the number specified is printed. Only the least significant 7 bits of the number is interpreted. *xx* is parsed according to the current radix.

Example

ADES-CLI> **W** ***Testing Completed.***

Results in this message being printed out.

XEQ

Syntax

[Xeq] *filename[.[prg,togl]] [passcount [start_adr]]*

Description

Loads the specified file into memory starting at physical location 0, and executes it. If a file type is not given, then a search is made, first through ADES program files, then toggle program files. If you indicate a file type, then the search takes place only on that type.

The pass count is the number of passes a program makes before returning to ADES. If you do not specify a count, the default *PASMAX* is used (as defined by the **DEFAULT** *PASMAX* command). The initial default PASMAX is 1.

If you don't give a start address, program files are started at location 200 (8); toggle files are started at location 0.

When executing a DTOS program, a pass count of -1 causes the program to run in a DTOS manual mode; pass counts other than -1 cause the program to run in a DTOS automatic mode for the number of passes specified.

You can leave out the **XEQ** portion of the execute command and simply type in the filename along with any arguments.

Example

ADES-CLI> **X** *MITCH.PRG 3*

Loads the program file, *MITCH*, into memory and executes it; also changes the number of passes the program makes to 3.

Section 4
ADES Utilities

Chapter 1
The Symbolic Debugger (ADEB)

The Symbolic Debugger (ADEB) lets you debug programs by accessing and changing the contents of any internal register or memory location. You can do this by controlling a program's execution with breakpoints (which stop the program at specific points). By placing breakpoints at appropriate places in a program, you can monitor a test program's progress and examine the operation of subtests. The Debugger lets you insert up to eight different breakpoints in a program. After a test program halts at a breakpoint, you can examine or change the values in memory or in certain registers. Then, you can restart the test program. Breakpoints defined with ADEB are maintained until you instruct ADEB to remove them.

When you run the Debugger, the CLI loads it first, followed by the ADES test program. If the program is too long to fit, you will not be able to run it with the Debugger and you will get the message MEMORY RESOURCES NOT AVAILABLE. (ADEB requires 2.5 KW. On a 16 KW machine you have 5.5 KW available to you; with 32 KW or more, you have 21.5 KW.)

Also note that ADEB uses locations 50-57 in memory, so be sure your test programs do not use these locations.

Executing the Debugger

To execute the debugger, type:

ADES-CLI> **DEBug** *filename*[.PRG] *[start_address]*

where *filename* is the name of the program you are loading. If you provide an extension other than .PRG, the CLI issues an error message. *Start_address* is the optional starting address for the Debugger. If you do not provide a starting address, the Debugger is loaded at the top of user space in memory.

For example,

ADES-CLI> **DEBUG** *PROG1.PRG*

loads the Debugger at the top of user space in memory and loads *PROG1* at location 0.

The CLI then prints the starting address of the Debugger and starts it. Be sure to type commands in uppercase; ADEB does not recognize lowercase letters.

Table 1.1 summarizes the basic debugger commands. For a more complete discussion, refer to the *Symbolic Debugger User's Manual* (093-000044) or the *ECLIPSE Symbolic Debugger* (093-000140).

NOTE: The Debugger displays a ? or U when you use an invalid command.

Table 1-1. Debugger Commands (continues)

Operation	Command	Function
Display/change memory cells	adr! adr/	Open location adr Open location adr and print contents
	CR	Close open location and open next one
	LF	Close open location
	^	Close open location and open preceding one
Display/change registers	$A	Display contents of all accumulators
	n$A	Open register n (n = 0-3 for the NOVA computer; 0-7 for the Eclipse computer)
Display/set breakpoints	$B	Display location of all set program breakpoints
	adr$B	Insert breakpoint at location adr
Delete breakpoints	$D n$D	Delete all breakpoints Delete breakpoint n (n = 0-7)
Restart program execution	$P	Restart execution from a breakpoint with break proceed counter set to +1 (executes breakpoint once before entering debugger program)
	n$P	Restart execution from a breakpoint with break proceed counter set to counter n (n = 0-7)
	n$Q	Open break proceed counter n (n = 0-7)
	adr$R	Restart execution at adr with I/O reset
Select output mode	=	Display output in numeric format
(use as terminator for input or after display of cell contents)	:	Display output in symbolic format
	;	Display output in instruction format
	<-	Display output in half-word format
	'	Display output in ASCIII format
	&	Display output in byte pointer format

Table 1-1. Debugger Commands (concluded)

Operation	Command	Function
Set output mode	$=	Display future output in numeric format
	$:	Display future output in symbolic format
	$;	Display future output in instruction format
	$<-	Display future output in half-word format
	$'	Display future output in ASCII format
	$&	Display future output in byte pointer format
Search memory (press any key to stop search)	$S	Search all memory
	adr$S	Search memory from location 0 to adr
	adr<$S	Search memory from adr to memory limit
	adr1<adr2$S	Search memory from adr1 to adr2
Display/change	$C	Open carry/teletype output register
Special registers	$F $L $M $N $W	Open floating point registers* Open location register Open mask register Open number register Open word register

*Applies to systems with floating point option

Chapter 2
The ADES Media Builder (AMB)

The ADES Media Builder (AMB) is a utility for creating new system media from runtime media. You have the option of making an exact replica of the runtime media or of selecting certain files from which to build.

The ADES Media Builder can create a system only on scratch media or ADES media; that is, one which is not write-protected or which already has ADES software on it. Check on the status of the media with the **EQUIPMENT** command. If the media is write-protected, change its status either with the **UNPROTECT** or **SCRATCH** commands or with the Manual Sizer (MSIZE).

The Media Builder operates in two modes: automatic and manual.

NOTE: The Media Builder automatically creates permanent files.

Automatic Mode

Use automatic mode if you want to include all the files which make up the runtime media in the media you are building. If you would have taken all the defaults in manual mode, run in automatic mode to save yourself time. You have the added advantage of being able to execute the automatic mode of the media builder from a macro script.

In automatic mode, the Media Builder builds a system on the first scratch media it finds in the System Equipment Table (the table is ordered by device code and unit number). The Builder copies all the files from the runtime media to the build device, unless instructed otherwise (see *Customized Building* below). If you want to control which device the Media Builder uses, you must make sure that all the devices ahead of it in the SET have write protection.

Before executing the Media Builder in automatic mode, set the switch register (SWREG). This is a 32-bit register which allows you to pass information to programs. (See Table 2.1 for a summary of all the bits in SWREG.) To run in automatic mode, set bit 15 to 1 and bits 16-31 to the model number of the device on which you are going to boot the media you are building.

To do this, use the **SWREG** command, followed by two arguments: a 1 and the model number. For example, to build on a 6045 disk drive, you would type:

```
ADES-CLI> SWREG 1 6045.
```

Notice the period (.) after 6045, indicating that the model number is in decimal, not octal.

To execute the Media Builder, type:

```
ADES-CLI> XEQ AMB
```

or,

```
ADES-CLI> AMB
```

When the Media Builder finishes, it then gives you a message telling you it has built the media.

Table 2-1. SWREG Summary

Bits	Description
0-7	Unused.
8	0 = Normal build. 1 = Customized build.
9	0 = Customizes the build to the current equipment table if STRING is null; or uses the equipment table in the UDF file specified in STRING. 1 = Customizes the build by including only those files listed in the TXT file specified in STRING. Bit 9 is only applicable if bit 8 = 1.
10	Reserved for future use.
11	0 = Unknown disks and tapes are write-protected. 1 = Unknown disks and tapes are write-enabled or scratch.
12	0 = Installs a new system bootstrap and does a surface analysis on the media (initializes the media). Any files currently on the media are destroyed. All selected files are transferred to the media. 1 = Does not install a new system bootstrap; uses the free file space currently defined on the media. Those files already on the media are not destroyed. Only new files are appended to the media. Bit 12 is only applicable if Bit 15 = 1.
13	0 = Reports block number of unusable sector and restarts disk build. 1 = Aborts disk build upon finding an unrecoverable unusable sector.
14	0 = Does not log transferred files to the system console. 1 = Logs transferred files to the system console.
15	0 = Manual mode of operation (questions asked). 1 = Automatic mode of operation (no questions asked).
16-31	If Bit 15 = 1, then Bits 16-31 contain the boot media model number.

Manual Mode

If you want to select files for the new media or if you plan to build several media in one sitting, use the Media Builder in manual mode. Refer to Figure 2.1 for a flowchart of media building in manual mode.

Before executing the program:

1. Make sure that bit 15 of the switch register is set to 0, otherwise the builder will assume you want automatic mode.

NOTE: SWREG is always cleared when a program returns, when an abort sequence is executed, and when ADES is brought up.

2. If you want tracking information to appear as the program is running, set bit 14 of the switch register to 1. By doing this, you will see which files have been transferred. To set the switch register, type:

   ADES-CLI> **SWREG** *2*

3. If you take the second option, we suggest you turn off console logging with the **LOG** command to avoid wasting log space.
4. If you want the Media Builder to abort if it encounters an unusable sector, set bit 13 of the switch register to 1. If you set this bit and the Builder finds an unusable sector, you get an error message and the program aborts. If you do not set bit 13 and the Media Builder finds an unusable sector, the program builds around the unusable sector after informing you that it has found one.

Execute the Media Builder in the same way as in automatic mode:

ADES-CLI> **[XEQ]** *AMB*

The program begins by asking four questions:

1. BOOT MEDIA MODEL NUMBER?

   Enter the model number of the device on which you are going to boot the media you are building; for example, 6045. The program repeats this question if the model number you give is not supported or if the System Equipment Table does not contain a device on which you can build a media with that model number.

2. DEVICE CODE [xx]?

   Enter the device code of the drive on which the media is being built. The default is the first scratch media that the Media Builder finds in the System Equipment Table that is capable of building a media of the model number specified. (Just press NEW LINE if you want the default.)

3. UNIT NUMBER [xx]?

   Enter the unit number of the drive on which the media is being built. The default is the first scratch media found in the System Equipment Table that is capable of building a media of the model number specified.

4. SYSTEM I.D. [xxxxxxxx]?

   Enter the system identification message for the media being built. The default is the runtime media's identification message (for details, see the description of **SYSID** in Section 3).

The Media Builder now lists a choice of operations:

SPECIFY OPERATION (?,A,B,D,E,F,P) [B]?

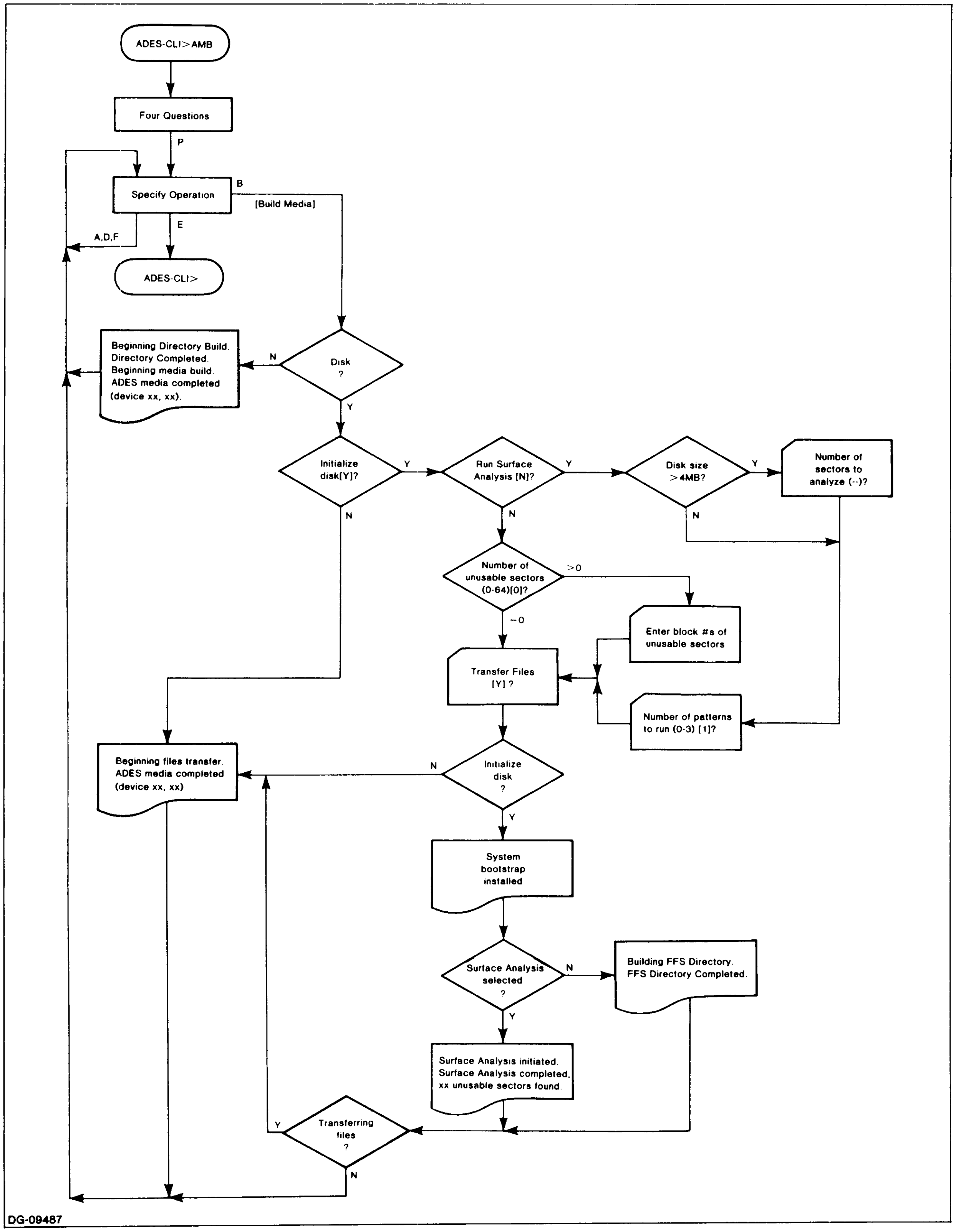
ADES-CLI>AMB
Four Questions
P
Specify Operation
B
[Build Media]
A,D,F
E
ADES-CLI>
Disk ?
N
Beginning Directory Build. Directory Completed. Beginning media build. ADES media completed (device xx, xx).
Y
Initialize disk[Y]?
Y
Run Surface Analysis [N]?
Y
Disk size >4MB?
Y
Number of sectors to analyze (--)?
N
N
N
Number of unusable sectors (0-64)[0]?
>0
=0
Enter block #s of unusable sectors
Transfer Files [Y] ?
Number of patterns to run (0-3) [1]?
Beginning files transfer. ADES media completed (device xx, xx)
N
Initialize disk ?
Y
System bootstrap installed
Surface Analysis selected ?
N
Building FFS Directory. FFS Directory Completed.
Y
Surface Analysis initiated. Surface Analysis completed, xx unusable sectors found.
Y
Transferring files ?
N
DG-09487

Figure 2-1. Manual Mode Flowchart

The default is to build the media. Table 2.2 summarizes the commands.

Table 2-2. AMB Commands in Manual Mode

Command	Meaning
?	Briefly explains the options
A	Adds files to the list that tells the Media Builder which files to copy from the runtime media to the new media
B	Builds the media
D	Deletes files from the list that tells the Media Builder which files to copy from the runtime media to the new media
E	Exits from the Media Builder and returns to the CLI
F	Indicates whether a file is being copied or not
P	Redefines the parameters for the new media (i.e., asks the initial four questions again)

The ? Command

Use the **?** command for a brief explanation of the choices available to you.

The A, D, and F Commands

In manual mode you have the option of copying selected files by manipulating the selection list. Initially, this list includes all the files on the runtime media (unless you are using the Custom Building option). You may, however, use the **D** command to delete files from the list.

When you use the **A**, **D**, or **F** command, you can refer to files directly by name or use the * and + templates as short-cuts. The * matches single characters; the +, any number of characters. Templates that do not contain a # prefix refer only to user files; those with a # prefix refer to system files. (Note that the # is not parsed as part of the filename.) Refer back to Section 1, Chapter 5, "The File System."

The system prompts you for specific file information with the message: SPECIFY FILENAMES/TEMPLATES? You can answer this question with one or more filenames and/or templates, separated by commas.

You can choose to delete, or inhibit, either system files or user files. (Even though system files are required by ADES in order to run, some are utilities that you may not need on the media you are building.) If you use templates, be sure you do not inadvertently delete any system files that you need. The **D** command confirms its actions by telling you that it has inhibited a file.

NOTE: If you inhibit a system file, the AMB will ask you for a confirmation before allowing you to build the media.

Also, be aware that files are deleted by families. A family consists of a parent file, such as a program, plus any subordinate files that the parent requires, such as overlay files. As a result, you need refer only to parent files; their families are included automatically. Note that some subordinate files are shared by several parent files. If you delete a family which contains a shared subordinate, this file is not deleted if it is still required by some other family. To see which files are parents and which are subordinates, use the **F** command, described below.

If you change your mind about deleting certain files, you can use the **A** command to add them back to the selection list. Again, you can either use filenames or templates when referring to files. The **A** command confirms its actions by telling you that it has enabled a file.

You can find out the status of a file with the **F** command. You can see all your files by giving the **F** command and a + argument or you can name the files about which you want information (templates are allowed). When you use the **F** command, it first lists those files which are enabled and then those which are inhibited. It further distinguishes between system and user files: system files are preceded with *. Also, subordinate files are indented, to indicate that they are part of a family. For example:

```
  PROG1.PRG
        ULIB0.OLF
        ULIB2.OLF
* ACOMP.PRG
  PROG3.PRG
        ULIB1.OLF
        ULIB2.OLF
```

means that ULIB0 and ULIB2 are overlay files that are part of the PROG1 family. Notice how ULIB2 is also part of the PROG3 family, making it a shared subordinate. The asterisk before ACOMP.PRG identifies it as a system file.

The B Command

Once you have selected which files you wish to copy, you can build the media with the **B** command.

NOTE: If you choose the build operation when building to a disk, and then change your mind, simply press the ESC key in response to any question (you will see a $ on the screen), and you will return to the SPECIFY OPERATION question; no action is taken on the operation you initially chose.

If any system files are inhibited, a confirmation is required before building the media:

```
SYSTEM FILE OR FILES INHIBITED. CONFIRM (Y,N) [N]?
```

To build the media, answer **Y**; otherwise, you will return to the Selective Builder prompt.

When you build a media on tape, the program issues messages informing you of its progress:

BEGINNING DIRECTORY BUILD.
DIRECTORY COMPLETED, BEGINNING MEDIA BUILD.
ADES MEDIA COMPLETED *(device xx,xx)*.

When you build on disk, you go through several question and answer steps. The sequence of questions and messages can be seen in Figure 2.1, Manual Mode Flowchart.

1. The AMB begins by asking:

   ```
   INITIALIZE DISK [Y]?
   ```

   If you answer yes, the program destroys any data that was previously on the disk and installs a new system bootstrap. You then have the choice of running a surface analysis on the disk.

   If you answer no, the Builder assumes that the disk has been initialized and automatically transfers the files. You can then choose another AMB command.

2. If you initialized the disk, the AMB asks:

   ```
   RUN SURFACE ANALYSIS [N]?
        TRANSFER FILES [Y]?
   ```

   If you choose not to run surface analysis, you must answer the questions in If you do choose to run it, answer the questions in 3. You should answer No to the transfer files question only if you are using the Builder to initialize disks.

3. If you are analyzing the surface, and the capacity of the disk is greater than 4 MB, the AMB asks:

   ```
   NUMBER OF SECTORS TO ANALYZE (MIN= 1000, MAX= xxxxxx) [1000]?
   ```

   You must supply the number of sectors to analyze if the disk's capacity is greater than 512 KB. You must also choose the number of surface analysis patterns to run.

   Analyze more than 8192 sectors only if you expect to use more than 8192 sectors of the build media. Regardless of disk size, the builder then asks:

   ```
   NUMBER OF PATTERNS TO RUN (0-3) [1]?
   ```

   Note that if you choose 0 patterns, the analysis is based only on the bad hardware sector flag of each sector, if it exists. If you answered yes to the initialize disk question, then the transfer file question is asked here.

   Assuming you selected initialization, surface analysis, and files transfer, the output would appear as follows:

   ```
   SYSTEM BOOTSTRAP INSTALLED.

   SURFACE ANALYSIS INITIATED.
        SURFACE ANALYSIS COMPLETED, xx UNUSABLE SECTORS FOUND.

   BEGINNING FILES TRANSFER.
        ADES MEDIA COMPLETED (DEVICE xx, xx).
   ```

   If, during the surface analysis process, you want to see how many sectors still have to be analyzed, press the ESC key. AMB provides the answer in the form of a 32-bit decimal number.

   When AMB is transferring files, if you did not select file logging by setting

SWREG bit 14, hitting ESC will cause AMB to print the file index number and name of the last file transferred to the build media. Also, a CTRL-C-ESC causes the remainder of the files NOT to transfer. The message FILE TRANSFER ABORTED is printed.

NOTE: System response will not be immediate.

4. If you chose not to analyze the surface (step 2), the AMB asks the following:

```
NUMBER OF UNUSABLE SECTORS (0-64) [0]?
     (1)? xxxxxx
     (2)? xxxxxx
      :    :
     (xx)? xxxxxx
```

The last four lines indicate that when you enter the number of unusable sectors, the AMB then asks you the block numbers of the unusable sectors. (This information would be available to you if you had previously initialized and/or built to the build media.) If you do not know of any unusable sectors, use the default answer of 0.

If you selected initialization without surface analysis, the surface analysis messages would be replaced with:

```
BUILDING FREE FILE SPACE DIRECTORY.
     FREE FILE SPACE DIRECTORY COMPLETED.
```

If you requested tracking by setting bit 14 of the switch register, you will also see the index numbers and names of all the files that are being copied to the new media.

Multiple Media Building

When the space required to hold all of the selected files exceeds the capacity of the build media, the AMB will continue its build on another media if the media is a removable disk. This process is called *multiple media building.* During multiple media building, system files are repeated on every media. For user files, the AMB takes care not to break up families when deciding which files go on which media. For example, if a program exists on a media built by the AMB, it is guaranteed that all of its subordinate files are also on the media.

The E Command

Use the **E** command to exit from the Media Builder and return to the CLI.

The P Command

The **P** command is very useful if you are building several media in one sitting, as the command results in the AMB asking the four questions about model number, device number, unit number, and system ID (see above). You will find that the defaults are now those of the media you last built. You can change all or some of the parameters.

Customized Building

The AMB can build a disk with a customized set of files. The list of files transferred can be customized via the equipment table or a user-specified files list.

When you customize via the equipment table, it selects the system files and all ADES programs that test equipment listed in either the current System Equipment Table, or an equipment table saved in a UDF file. To customize via an equipment table, you must set SWREG bit 8 and clear SWREG bit 9. If you are customizing to the current System Equipment Table, then clear the STRING buffer too. To customize to an equipment table saved in a UDF file, enter the name of the file in the STRING buffer. The build is aborted if the file you specify does not exist.

When you customize to a user-specified files list, only those files that are listed in a specified TXT file are selected. To customize via a user-specified files list, you'll have to set SWREG bits 8 and 9. Then, enter the name of the TXT file that contains the files list in the STRING buffer. The builder ignores any files in the user-specified files list that do not exist on the runtime media. See Figure 2.2 for an example of a Custom Files List.

```
CCMSZ.RYSS
MHMINI.RSYS
KERNEL.RSYS
SINIT.RSYS
ASIZE.PRG
MSIZE.PRG
CSMOD1.OLF
CSMOD2.OLF
CSMOD3.OLF
SPMON.OLF
ERMSG0.SDF
ERMSG1.SDF
ERMSG2.SDF
ERMSG3.SDF
ERMSG4.SDF
ABORT.SNRU
BYE.SNRU
CHAIN.SNRU
CLI.SNRU
CLIERR.SNRU
CLISR.SNRU
ECLASS.SNRU
EQUIPMENT.SNRU
ERRMSG.SNRU
EXCCHR.SNRU
FINDEV.SNRU
GETDRV.SNRU
ONEK.SNRU
SEDIT.SNRU
SWREG.SNRU
TERMINATE.SNRU
TTYIN.SNRU
```

Figure 2-2. Custom Files List (continues)

```
WRITE.SNRU
XEQ.SNRU
MH_DRIVER.SNRU
TTY_DRIVER.SNRU
SPVL.OLF.0
SPVH.OLF.0
SPMC.OLF.0
SEARCH.PRG.4
MHODRV.DRVR.0
ENDB.UDF.0
ARI.SRCH.0
CSP.SRCH.0
DCU.SRCH.0
DPP.SRCH.0
IAC.SRCH.0
IPP.SRCH.0
ISP.SRCH.0
LMP.SRCH.0
LPP.SRCH.0
MCP.SRCH.0
MDV.SRCH.0
MEM.SRCH.0
MHP.SRCH.0
MTP.SRCH.0
NFP.SRCH.0
PGP.SRCH.0
PTP.SRCH.0
SCM.SRCH.0
TTS.SRCH.0
ZBP.SRCH.0
GENMN.OLF.0
GMON.OLF.0
CLI.UDF.0
```

Figure 2-2. Custom Files List

Chapter 3
The Edit Utility

The Edit utility lets you edit any type of existing file and create source, text, or toggle files.

Edit not only serves as a text handling utility, but also as a patch utility for putting fixes in programs. If you are editing a nontext file, the data in the file appears in numerical form (in the current radix). Note also that when working with source or toggle files, you can execute them right from the Edit utility.

NOTE: You cannot execute Edit within a script.

The Edit utility numbers the lines in a file for your convenience (the numbers are not part of the file). If you are working with ASCII text, the line numbers are decimal; otherwise, they are in the current radix. Most Edit commands work with one entire line or a group (range) of lines. As you modify a file, the line numbers are automatically updated to reflect additions or deletions.

When you edit a file, you move the cursor from line to line. The current line is the one on which the cursor is currently sitting. This is the line that you last displayed, modified, inserted, or appended; when you start editing a file, the current line is the first line of the file.

Control characters that are either entered on the comand line or exist in the text file being edited are displayed on uppercase LCD terminals as blinking characters, and are displayed as dimmed characters on upper/lowercase LCD terminals, graphic displays, and color displays. However, the **LIST** command prints the control characters literally (e.g., a <14> will cause an ERASE PAGE character to be printed instead of a blinking or dimmed "L").

If you are on a terminal that supports dimmed characters, but prefer to see the control characters blinked instead of dimmed, you can resize (via the Manual Sizer) the system console as an uppercase LCD terminal (model 6052). If you are on an uppercase LCD terminal, and do not want the control characters blinked at all, type CTRL-P CTRL-D to disable blinking (CTRL-P CTRL-C reenables blinking).

Executing Edit

Before executing Edit, make sure the OPERATOR flag is on (see **OPERATOR** in Section 3). Then type:

ADES-CLI> **EDit** *filename.filetype*

where *filename.filetype* is the name and extension of the file you wish to create or edit. If you omit the filename and/or extension, Edit prompts you for it (for a review of extensions, see Section 1, Chapter 5, "The File System").

If the file exists, you get the Edit prompt, EDIT-CLI>.

If you indicate you want to work on a source (SRCE), text (TXT), or toggle (TOGL) file, and the file does not exist, Edit asks you:

CREATE NEW FILE (Y,N) [N]?

If you answer yes, you get the EDIT-CLI prompt; otherwise you get the ADES-CLI prompt.

Once you have the Edit prompt, you can enter any Edit command. We will now discuss the commands in detail.

Commands

Table 3.1 summarizes all the Edit commands.

Table 3-1. Summary of Edit Commands

Command	Meaning
ABORT	Discards all changes and returns to ADES-CLI
APPEND	Appends lines to file
BYE	Saves the file on runtime media and returns to ADES-CLI
DELETE	Deletes one or more lines
EXE-CUTE	Executes the source or toggle file
INSERT	Inserts text into the file
LIST	Displays one or more lines of the file
MODIFY	Modifies one or more lines of the file
SAVE	Saves the file on the runtime media and returns to ADES-CLI
STATUS	Displays information about the file being worked on

ABORT

Syntax

ABort

Edit then asks:

```
DO YOU REALLY WANT TO DISCARD YOUR EDITS (Y,N) [N]?
```

Answer yes or no. The default is no.

ABORT discards any changes that you made and returns you to ADES CLI. If the file you were working on was created during the Edit session, the file is deleted. When you use this command, Edit asks you to confirm your action. If you do not confirm, you get another Edit prompt.

APPEND

Syntax

APpend

APPEND adds (appends) lines to the end of the file you are editing. To add lines in the middle of the file, use the **INSERT** command.

After you type in the **APPEND** command, an asterisk appears before the line number to indicate that you are in append mode. Type in the lines you wish to add, concluding each line by pressing NEW LINE. When you finish, press ESC to get back the EDIT-CLI prompt.

BYE

Syntax

Bye *[filename]*

BYE saves the file being worked on and returns you to ADES CLI. This command is identical to Edit's **SAVE** command.

DELETE

Syntax

Delete *[ALL]*
[LAST]
[line_number [line_number]
[LAST]]

ALL means the entire file; *LAST*, the last line of the file. Note that you can provide two line numbers to indicate a range (that is, a block of lines) or a line number and the word *LAST* to indicate a block from some line in the file to the last line.

NOTE: Remember to provide line numbers in the current radix; you may use a radix indicator.

DELETE gives you the choice of deleting the current line, all the lines, the last line of the file, a specific line, or a range of lines in a file.

Examples

EDIT-CLI> **D**

Deletes the current line.

EDIT-CLI> **DELETE** *ALL*

Deletes the entire file.

EDIT-CLI> **DEL** *LAST*

Deletes the last line of the file.

EDIT-CLI> **D** *3*

Deletes the third line.

EDIT-CLI> **D** *3 7*

Deletes lines 3 through 7.

EDIT-CLI> **DELETE** *21 LAST*

Deletes from line 21 to the end of the file.

EXECUTE

Syntax

Execute *[togl_start_address]*

togl_start_address is the starting address for the toggle file. (If you provide an address for a source file, you will get an error.)

If you try to execute a file other than a source or toggle file, you get an ILLEGAL FILE TYPE message and an Edit prompt. If the file is empty, you get an EMPTY FILE message and an Edit prompt.

If everything is correct, however, you are asked to confirm execution:

```
CONFIRM (Y,N) [N]?
```

Yes results in execution. No gets you another Edit prompt. You cannot return to ADES from the toggle program execution (you must manually reboot the media). When the script completes, you are placed back in ADES-CLI.

This command executes a toggle file or a one-page (256-word) source file. If the source file is longer, you will get the message MEMORY RESOURCES NOT AVAILABLE. Also, this macro file may not call other macros. If the edit file is a toggle file, the program you execute is moved to location 0. Execution starts at location 0 unless you provide a starting address.

NOTE: Any changes you made during the Edit session are not saved. To save your edits, use the SAVE (or BYE) command.

INSERT

Syntax

Insert *[line_number]*
[LAST]

Without an argument, **INSERT** inserts before the current line. Otherwise, you can indicate a specific line with *line_number* or the last line with *LAST.*

INSERT lets you insert text before the current line, another specified line, or the last line. **INSERT** lets you add up to 256 words (512 characters) in one operation. If you exceed this limit, you get an error message saying INSERT BUFFER FULL.

On a 32 KW (or greater) machine, you can have files up to 21.5 KW; on a 16 KW machine, your files can reach 5.5 KW.

When you use **INSERT**, you enter insert mode, indicated by a * prompt. This is followed by a line number (for text files) or a word address (for other files). Type in the data you wish to insert (remembering the limit). To exit from insert mode, use ESC for text files and any non-numeric character, NEW LINE, or Carriage Return for other files.

Examples

EDIT-CLI> **I** *20*

```
*20 This text is inserted before line 20.
*21 $
```

EDIT-CLI> **I** *LAST*

```
*xxx This text is inserted before last line in file.
*xxx $
```

LIST

Syntax

```
List  [ALL]
      [LAST]
      [line_number   [line_number]]
                     [LAST]
```

ALL means the entire file; *LAST*, the last line of a file; and *line_number*, a particular line, referred to by line number. Line numbers apply only to text files; for other files, provide the relative word address. Note that you can provide two line numbers to indicate a range (that is, a block of lines) or a line number and the word *LAST* to indicate a block of lines from a spot in the file to the end of it.

The **LIST** command lets you display the current line, the entire file, the last line of a file, a specific line, or a range of lines.

Examples

EDIT-CLI> **L**

Displays the current line.

EDIT-CLI> **LIST** *ALL*

Displays the entire file.

EDIT-CLI> **LIST** *LAST*

Displays the last line of the file.

EDIT-CLI> **L** *3*

Displays the third line.

EDIT-CLI> **L** *3 7*

Displays lines 3 through 7.

EDIT-CLI> **LIST** *21 LAST*

Displays from line 21 to the end of the file.

MODIFY

Syntax

```
Modify  [All]
        [Last]
        [adr [adr]]
             [Last]]
```

All is the entire file; *Last* is the last word of the file; *adr* is the specified address (or range of addresses); and *adr Last* is the range of addresses between *adr* and the last word of the file.

MODIFY allows you to modify lines in a source file, only when the system console is a video device. When this is the case, you can modify the lines using the Screen Edit Utility. The prompt is the line number followed by the source file line. You can exit Modify mode prior to the last line specified by typing ESC.

You can modify lines in numeric files regardless of console type. You will get a prompt that is the word address, followed by the current value at that address. You then type in the new numeric data, followed by NEW LINE. If you type in a non-numeric character while entering numeric data, you leave Modify mode and return to EDIT-CLI.

SAVE

Syntax

SAve *[filename]*

Saves the file on the runtime media; returns to ADES-CLI.

If a filename is not provided, or if a filename is provided and it is the same as the filename specified when the Edit utility was invoked, then the filename provided with the **EDIT** command is used.

If a filename is provided on the **SAVE** command that differs from the one specified on the **EDIT** command, then if a file was created when the Edit utility was invoked, it is renamed to the filename specified on the **SAVE** command. If the file specified on the **EDIT** command already existed, then specifying a new filename on the **SAVE** command gives the operator the option of deleting the old file.

The **SAVE** command is ignored if there isn't enough space for the edited file on the runtime media and a warning is issued: FILE SPACE EXHAUSTED. If the file is empty, EMPTY FILE appears, and a new EDIT CLI prompt is issued.

NOTE: You cannot use **SAVE** if the runtime media is a tape; use **EXECUTE** or **ABORT** to exit from EDIT when running off tape.

STATUS

Syntax

STatus

Gives the following information on the file: name, type, current line number or word address, last line number or word address, number of 256-word pages in edit file, whether the file is empty, and/or execute only, if applicable.

Example

```
EDIT-CLI> ST
```

results in the following:

```
FILENAME: xxxxxxxxx      FILE TYPE: xxxxxxxxx
CURRENT LINE: xxxx       LAST LINE: xxxxx

NUMBER OF PAGES: xx
EMPTY FILE:
EXECUTE ONLY FILE:
```

The last two messages are printed only when they apply.

Chapter 4
The Octal Debugging Tool (ODT)

The Octal Debugging Tool (ODT) provides some limited debugging facilities. The advantage of using the ODT rather than the Symbolic Debugger (ADEB) is that the ODT does not use any ADES system routines. Therefore, it doesn't have to rely on the ADES software in order to operate. The ODT, however, can be executed only from a primary teletype (device code 10/11), an MBC (device code 20), or an MPT memory-mapped screen. (In the last case, ADES system routines are used.) Breakpoints defined with the ODT are automatically removed once the breakpoint is entered.

You can also use the ODT for system restarts (if you can access the ODT at location 202) and memory dumps, as explained under the M command below.

The prompt for the ODT is the at-sign (@). When you see this sign, the ODT is ready for you to input commands. If you type in an illegal command or expression, the ODT prints a question mark and gives you a new prompt.

Access to ODT

The ODT can be accessed in four ways: by using breakpoints; by issuing a system call command; by jumping to location 202 or starting a hard/soft console at location 202; and finally, by a system panic.

When breakpoints are used, the **P** command will continue execution at the breakpoint address. The system call command format is CTRL-O [SCALL ODT]. If you access the ODT in this manner, the **P** command returns to call +1. After accessing ODT at location 202, the **P** command does a system restart. (A restart involves clearing key system variables, executing an abort (CTRL-C CTRL-B) sequence, and issuing a new ADES-CLI prompt.) If the ODT is accessed through a system panic, the **P** command will do a system restart.

Command Formats

ODT commands consist of either a letter, or an expression followed by a letter. An expression is one or more numbers separated by a plus (+) or minus (-) sign. The period symbol (.) can be used in place of a number anywhere in an expression. The value of the period symbol is equal to the total value of the last expression used.

Commands

When using the ODT commands, you will provide an expression or a number to open either a memory location or an accumulator, respectively. For example, typing in the expression 100 + 23 would open a particular memory location. When a memory location or accumulator is opened, its contents are printed out. After the contents are printed you may alter them by typing in a new number. When you close the location, you store the typed-in number in the memory location or accumulator you previously opened.

To open an accumulator, you must provide a number from 0-14, followed by the letter A. Each number represents a specific accumulator as indicated below.

WARNING: Do not open any accumulators other than the ones you see in Table 4.1.

Table 4-1. Accumulator Numbers

Number	Accumulator
0	AC0
1	AC1
2	AC2
3	AC3
4	Carry
5	ION flag 0 = interrupts were off -1 = interrupts were on
6	Subtotal of current expression
7	Entry flag -1 = CTRL-O 0 = breakpoint
10	Dump media device code
11	Dump media unit number
12	Contents of location indicated by software frame pointer
13	Total of last expression (.)
14	Proceed address — do not modify this location

The remainder of the ODT commands are used to open and close memory locations and accumulators, to insert breakpoint instructions, to delete breakpoints, to start execution of a program at a certain location, and to perform memory dumps. Table 4.2 summarizes the rest of the ODT commands.

NOTE: The ODT accepts either uppercase or lowercase letters for alphabetic commands.

Table 4-2. ODT Commands

Command	Function
expression/	Opens the memory location specified in expression.
NEW LINE (key)	Closes the currently opened memory location or accumulator; a new prompt is issued.
Carriage Return (key)	Closes the currently opened memory location or accumulator and opens the next location or accumulator.
Up Arrow (key)	Closes the currently opened memory location or accumulator and opens the previous location or accumulator.
/	Closes the currently opened memory location or accumulator and opens the memory location that is specified as the contents of the location or accumulator most recently closed.
expression B	Places a breakpoint instruction at location specified. When the specified instruction is executed, the ODT is re-entered and the actual instruction is restored.
B	Lists the location of the breakpoint defined by expression B command.
D	Deletes the breakpoint defined by the expression B command (in other words, restores the actual instruction at the breakpoint address).
P	Proceeds according to the way the ODT was entered.
expression R	Starts execution at nonzero expression specified.
I	Performs an IORST instruction. Do not execute this instruction if the system console or the secondary output device is on a multiplexor.
expression=	Prints the value of specified expression and places in accumulator 13.
M	Dumps all logical memory to magtape on specified device code and unit number, which are always 10A and 11A, respectively. Operator must mount a scratch tape and confirm the memory dump. You return to the ODT when you remount the original tape and press any key. Do not perform memory dumps while console logging is enabled. If you use the P command after a dump, this will result in a system restart.

Chapter 5
The Script Builder (SCRBLD)

The ADES Script Builder (SCRBLD) is used to generate a list of programs that run on the current CPU, test equipment that is listed in the SET, and that can be executed without asking any questions. On a tape, these programs are in the form of a memory-resident script that is executed and lost when the script finishes. On a disk, however, you have a choice. The programs can be in the form of a memory-resident script, or they can be in the form of a source file that contains a series of execute commands which can be saved on the runtime disk for later use.

Both files contain commands to test one or more parts of a system with a series of programs found on the runtime media. The Script Builder has three sources of information:

1. A cross-reference table that indicates which programs test which part of the system, and on which CPU they run. This table is supplied by Data General; you should NEVER modify it unless instructed.
2. The System Equipment Table
3. A directory structure that lists which programs are on the media. This directory structure is built into the media when it is created. (Note that the structure is affected when you delete, add, rename, or update programs.)

Executing SCRBLD

You can execute the Script Builder by typing in SCRBLD while in ADES-CLI. SCRBLD is also executed automatically from the RUN and ACCEPT macros. To run the Script Builder automatically, first turn off the OPERATOR flag with the **OPERATOR** command (see Section 3). Then type:

```
ADES-CLI> XEQ SCRBLD
```

or,

```
ADES-CLI> SCRBLD
```

A one-page script is built and immediately executed. It is based on the System Equipment Table that is currently in memory. The program then gives messages as follows:

```
BUILDING SCRIPT
SCRIPT BUILT
```

Once the script is executed, you return to the CLI prompt.

To run the Script Builder interactively, make sure the OPERATOR flag is on. Once the Script Builder is entered, you will be given a series of questions to answer, provided the OPERATOR flag is set.

To execute the program, type:

ADES-CLI> **XEQ SCRBLD**

or,

ADES-CLI> **SCRBLD**

You now are asked some questions:

1. If the runtime media is write-protected, only the question in step 3 is asked. Otherwise, the Script Builder asks:

   ```
   EXECUTE OR SAVE (E, S)?
   ```

   Answer **E** if you want to execute and discard a script; **S**, if you want to create and save a script source file. If you answer **E**, you go to step 3; if **S**, go to step 2.

2. If you indicated that you wanted to save the script source file, the Script Builder asks:

   ```
   SOURCE FILE NAME?
   ```

   Enter the name you want for the source file. You can use the name of an existing source file; the Script Builder will update it.

3. Finally, the utility asks:

   ```
   EQUIPMENT TABLE [CURRENT]?
   ```

   Press NEW LINE if you want to use the System Equipment Table currently in memory. Otherwise, enter the name of the file with whose contents you want to replace the SET. Figure 5.1 shows a sample interaction.

   If you set the VERIFY flag in the Environment Control Word to YES, you can run only best-bet programs, selected by the Script Builder.

   As you notice in Figure 5.1, after you answer the last question, the utility gives one of the following series of messages:

   If you EXECUTE:

   ```
   BUILDING SCRIPT FILE.
   SCRIPT FILE BUILT.
   ```

   If you SAVE:

   ```
   BUILDING SOURCE FILE.
   SOURCE FILE BUILT.
   ```

   You then return to the CLI prompt.

```
ADES-CLI> XEQ SCRBLD

ADES SCRIPT BUILDER REV X.X

EXECUTE OR SAVE (E,S)?  S

SOURCE FILE NAME?  TEST

EQUIPMENT TABLE [CURRENT]?

BUILDING SOURCE FILE.
SOURCE FILE BUILT.

ADES-CLI>
```

Figure 5-1. Sample Script Building Session

Chapter 6
Sizer Programs (ASIZE and MSIZE)

The sizer programs, ASIZE and MSIZE, let you set up or modify the System Equipment Table (SET). This is a table in memory which lists all the peripherals and options in the system. The SET includes information about the model numbers, device codes, and unit numbers of peripherals. The table also has such information as the type of the media and its write protection status. For disks, the dimensions are included. This kind of information is crucial to the diagnostic, reliability, and exerciser programs for peripherals.

To set up the System Equipment Table, you can use the Auto Sizer program, the Manual Sizer, or both. The Auto Sizer cannot size all peripherals completely (notably communications ones), and may produce erroneous information due to hardware errors or restrictions, so generally you will run both sizer programs so that the SET is complete and correct.

NOTE: The Auto Sizer does not save the information in the SET on the disk. To do so, use the **EQUIPMENT WRITE** command.

Auto Sizer (ASIZE)

You can run the Auto Sizer either when bringing up ADES (refer back to Section 1, Chapter 2, "Bringing Up ADES") or any time after you have a CLI prompt. The Auto Sizer (ASIZE) automatically sizes all the peripherals that it finds and adds the information to the System Equipment Table.

To execute the program, type:

ADES-CLI> **XEQ** *ASIZE*

or,

ADES-CLI> **ASIZE**

The output is a list of mnemonics and device codes that the listed mnemonics were sized on. For any device that it finds, the Auto Sizer adds an F after the device code. Figure 6.1 shows a sample list. Examine this list carefully to see whether the program is omitting any devices (you can also use the **EQUIPMENT** command to check the contents of the SET). If any peripherals are missing, use the Manual Sizer to enter information about them.

```
ADES-CLI> ASIZE

AUTOMATIC CONFIGURATION SIZER REV X.X

CPU IS ECLIPSE S/140

SIZE FOR NON-STANDARD DEVICE CODES [N]?  Y <NL>

ERCC     02F
MMPU1    03F
RTC      14F 54
PIT      43
PTP      13 53
PLT      15 55
CDR      16 56
COMM     75
FPU      76F
CHAR     --F
MTA      22F 62

ENTER NON-STANDARD DEVICE CODE (CR=DONE):

DKB     26 66 27 67 33 73

ENTER NON-STANDARD DEVICE CODE (CR=DONE):

SIZING DCU #64
IOT  00
ALM 34  44

ADES-CLI>

NOTE:  This is only a sample; your list will have more devices
sized for, and will have more devices allowing for non-standard
device codes.
```

Figure 6-1. Sample Auto Sizer Output (Incomplete)

Manual Sizer (MSIZE)

The Manual Sizer lets you manually add information about the system configuration to the SET. Generally, as we have pointed out, you will use the Manual Sizer (MSIZE) to correct the SET generated by ASIZE.

To run the manual sizer, type:

ADES-CLI> **XEQ MSIZE**

or,

ADES-CLI> **MSIZE**

NOTE: If the OPERATOR flag is off, the program issues an error and returns to ADES CLI. Use the **OPERATOR** command to turn OPERATOR on.

The Manual Sizer is self-documenting, so just follow the instructions given by the program. When you execute the program, you are asked a series of questions. These questions must be answered so that the program can enter the correct information. See Figure 6.2 for a sample output from MSIZE. To abort from MSIZE and return to ADES-CLI, type CTRL-C, CTRL-A.

Note the following:

1. If you hit ESC in answer to a question about device-dependent information, and you created the entry, then the entry is deleted and you get the ENTER DEVICE CODE question again.
2. You may only specify peripherals that run on the bus (processor) currently being sized (e.g., you cannot specify a 6095 (MicroProducts Phoenix Drive) if the processor being sized is a NOVA® or ECLIPSE)®.

```
ADES-CLI> MSIZE

MANUAL CONFIGURATION SIZER REV X.X

ENTER $ (ESCAPE) TO RESTART QUESTIONS.

CPU IS ECLIPSE S/140
PROCESSOR BEING SIZED IS NOVA/ECLIPSE

SPECIFY OPERATION (?,A,C,E,H,M,P) [A]: <NL>

ENTER DEVICE CODE [--]: 33

33      DKP
        MODIFY OR DELETE THIS ENTRY (M,D) [M]?
        UNIT #s TO ACCESS [0]:
          FOR UNIT #0
          MODEL # [6070]:
          MEDIA TYPE (?,A,L,N,P,R,S,U) [R]:
```

Figure 6-2. Sample MSIZE Output

Chapter 7
The Update Utility

The Update utility lets you update an ADES system disk without rebuilding it. This utility uses either an ADES update tape or an ADES system disk as a source of new (that is, updated) files. The source of the update is referred to as source media, and is used to update runtime media. The source may be disk or tape, but the runtime media may be only a disk.

Building Update Media

Although update media are distributed by Data General, you can create your own by using AOS **COPY** commands or RDOS **XFER** commands.

When you use AOS **COPY** commands, the first file on the tape must be the update directory, which defines the new System I.D. for the runtime media, and lists files on the update tape. The first line in the Update Directory contains the new System I.D.; the second line describes File 1; the third line describes File 2; and so on. The format of these lines is as follows:

filename.filetype major_rev.minor_rev

where the *major/minor* revision is in decimal.

Example:

```
NEADES_REV_00.01
MYFILE.PRG 3.0
MYSNRU.SNRU 1.0
```

The major/minor revision is in decimal.

When you copy SRCH files to the update tape, a record size of 4096 bytes must be used.

When you use RDOS **XFER** commands to build update media, you again have the update directory on File 0, and the lines that follow describe the files sequentially. The format of the lines is also the same as when you use AOS **COPY** commands, except that you end with filesize, as follows:

filename.filetype major_rev.minor_rev filesize

Filesize is the number of 256-word sectors of disk space the file needs. Since it is not possible to change the record size of RDOS formatted tapes, SRCH files cannot be updated using RDOS formatted update tapes.

Example:

```
NEADES_REV_00.01
MYFILE.PRG 3.0 3
MYSNRU.SNRU 1.0 2
```

The major/minor revision and file size are in decimal.

When using either AOS or RDOS, note that the new System I.D. must not exceed 79 characters, and must be terminated by a NEW LINE character (ASCII <12>).

Executing Update

Before executing the Update Utility, you must make sure that the source media you are using for the update is properly classified in th System Equipment Table. Using the Manual Sizer (MSIZE), add or modify the appropriate entry so that the classification is UPDATE (media type "U") if the source media is a tape, or UPDATE or ADES (media types "U" and "A", respectively) if the source media is a disk. Refer to Figure 7.1.

```
ADES-CLI>

ADES-CLI>  MSIZE

MANUAL CONFIGURATION SIZER      REV. 0.0

ENTER "$" (ESCAPE) TO RESTART QUESTIONS.

CPU IS ECLIPSE S/140
PROCESSOR BEING SIZED IS NOVA/ECLIPSE

SPECIFY OPERATION (?,A,C,E,H,M,P) [A]:  <NL>

ENTER DEVICE CODE [--]:  22  <NL>
DEVICE MNEMONIC:  MTA

22      MTA
        UNIT #'S TO ACCESS [0]:  <NL>
         FOR UNIT #0
          MODEL # [6020]:  <NL>
          MEDIA TYPE (?,A,L,N,P,R,S,U) [A]?  <NL>
        DATA CHANNEL MAPS (A,B,C,D) [A]?  <NL>

ENTER DEVICE CODE [--]:  $


CPU IS ECLIPSE S/140


PROCESSOR BEING SIZED IS NOVA/ECLIPSE

SPECIFY OPERATION (?,A,C,E,H,M,P)  [A]:  E  <NL>

SAVE NEW EQUIPMENT TABLE ON DISK (Y,N) [N]?  <NL>

PLACE NEW EQUIPMENT TABLE INTO MEMORY (Y,N) [Y]?  <NL>

ADES-CLI>
```

Figure 7-1. Listing an Update Tape in the System Equipment Table

Because Update is interactive, make sure the OPERATOR flag is on before executing the utility (see OPERATOR in Section 3). Execute the Update utility as follows:

ADES-CLI> **XEQ UPDATE**

or,

ADES-CLI> **UPDATE.PRG**

The utility now asks several questions; all the questions have default answers.

CAUTION: Be sure to use only NEW LINE or Carriage Return to terminate your answers. An ESC will abort the current Update utility prompt and return you to the CLI.

The questions are as follows:

1. DEVICE CODE [22]?

   Enter the device code of the source media. The default is 22. The System Equipment Table must define at least one of the units associated with the device code as an ADES update media (tape or disk based) or an ADES system media (disk based). Otherwise, you get an error message: EQUIPMENT NOT FOUND or NOT AN UPDATE MEDIA.

2. UNIT NUMBER [x]?

   Enter the unit number of the source media. The System Equipment Table must define the unit specified as ADES update media (tape or disk based) or ADES system media (disk based). Otherwise you get an error message: EQUIPMENT NOT FOUND or NOT AN UPDATE MEDIA. If the source media is a tape, you are asked question 3; otherwise, you go on to question 4.

3. FORMAT (0= AOS 1= RDOS) [0]?

   Enter the format used for the source tape.

4. CONFIRM [N]?

   If you take the default of No, Update just copies the update files to the runtime media. Otherwise, you must confirm the transfer of each file. Update will list the name of each file:

   filename.filetype [N]?

   and you must answer Yes, or press NEW LINE to answer the default of No, to indicate whether to copy the file over or not.

5. VERIFY [Y]?

   If you answer Yes, Update prints the name of each file it transfers to the runtime media. It includes the file type and the major/minor revision numbers. If you answer No, Update does not verify its actions.

6. APPEND [N]?

   If you answer No, Update transfers a file to the runtime media only if it already contains a file with that name. Question 7 is not asked.

   If you answer Yes, Update transfers a file to the runtime media even if no copy of it exists on the runtime media.

7. REPLACE [Y]?

   If you decided to append files, Update now asks whether you want to replace them. When you answer Yes, the Update utility replaces the files on the runtime media with files of the same name on the source media.

   If you answer No, Update transfers files to the runtime media only if they do not already exist on it. Question 8 is not asked.

8. IGNORE REVS [N]?

   If you are replacing files, you have the option of ignoring major revision levels. When you answer No, Update only transfers files if the major revision level of the source file is higher than that of the file on the runtime media. When you answer Yes, Update transfers files to the runtime media regardless of revision level.

Update now transfers files to the runtime media according to your instructions. When it finishes, you get the message:

--- UPDATE COMPLETE ---

and you return to the CLI prompt.

```
ADES-CLI> XEQ UPDATE

Device code [22]?
Unit # [0]?
Format (0=AOS, 1=RDOS) [0]?
Confirm [N]? Y
Verify [Y]?
Append [N]?
Ignore Revs [N]  Yes

MTBOOT_4KW.RSYS [N]?
AMB_TB.OLF [N]? Y
AMB_TB.OLF      REV X.X

  --- UPDATE COMPLETE ---
```

Figure 7-2. Update Interaction

Errors

If you get an EQUIPMENT NOT FOUND error message, you will return to ADES-CLI. If the OPERATOR flag is off when the Update Utility is executed, an OPERATOR NOT PRESENT error is issued, and again, you return to the CLI.

Chapter 8
ADES File Editor

The ADES File Editor (FEDIT.PRG) lets you display and patch any file on the runtime disk (media), and lets you generate toggle files. FEDIT is used primarily to patch files which exceed 24 KW in length, making them unaccessible to the Edit Utility. FEDIT also allows you to enter and display data as numbers, instructions, or ASCII characters.

When FEDIT is executed, you must first tell the utility which file to edit:

```
Pathname?
```

You must enter both the file's name and type (e.g., MYFILE.UDF). If there is any problem opening the file, the system issues an error message and repeats the question. If you wish to return to ADES-CLI at this point, press ESC.

If the file you request exists, it is opened and can be edited. If the file does not exist and the file type is not TOGL?, the system issues an error message and repeats the question.

If you requested a TOGL? file (e.g., MYFILE.TOGL) that doesn't exist, FEDIT lets you create one. The following question is asked:

```
CREATE A NEW FILE (Y,N) [N]?
```

If you don't want to create a new file, answer No and the pathname question is repeated. If you do wish to create a new file, answer Yes; the following question is asked:

```
FILE SIZE (# OF PAGES) [1]?
```

Enter the number of pages you want (note: one page equals 400 octal words). A toggle file of the specified size is created and then opened.

Once the file is opened, you enter the main loop:

```
Mode (?,N,I,A) [m] ?

Address (0 - xxx) [dadr]?

aaaaa xxxxxx yyyyyy<cr>
aaaaa xxxxxx yyyyyy<cr>
          :
          :
aaaaa xxxxxx yyyyyy<nl>
```

m is the current input/output mode (IOM). The three modes are: numeric (N), instruction (I), and ASCII (A). *dadr* is the default address, *aaaa* is the current address you are examining, and *xxxxxx* is the value at the current address, displayed according to the current IOM. If you want to change the value, type in the new value (shown here as *yyyyyy*), following the rules pertaining to the current IOM (see Table 8.1); otherwise, just type a delimiter. The delimiter you type controls what FEDIT does next. The following table summarizes the action each delimiter has.

Table 8-1. FEDIT Delimiter Actions

Prompt	Delimiter	Action
old value	<cr>	Close current location, open next location.
	<nl>	Close current location, return to the Address question.
	<esc>	Do not modify current location, return to Address question.
Address Question	<cr> <nl>	Open the address specified. Open the address specified.
	<esc>	Return to the Mode question.
Mode Question	<cr>	Define the input/output mode and go to the Address Question.
	<nl>	Define the input/output mode and go to the Address Question.
	<esc>	Close the file and return to ADES-CLI.

NOTE: The ^C^A and ^C^B abort sequences are disabled while FEDIT is running. This prevents the open file from being left in an unreadable state.

The current address is displayed in the current radix.

The input/output mode defines how the data at the current address is displayed and entered. The three modes and their syntax rules are as follows:

1. *Numeric format* displays and inputs data as 16-bit numbers in the current radix.
2. *Instruction format* displays and inputs data as either NOVA® or ECLIPSE® instructions. The CPU type on which you are currently running determines how the data is interpreted. If you are running on a NOVA® processor, all data is interpreted strictly as NOVA® instructions; if you are running on an ECLIPSE® processor, data is first checked to see if it is an ECLIPSE® instruction.

   The following rules apply when in Instruction input/output mode:

a. The last word in the file is always interpreted as a NOVA instruction. For example, in a 4 page file, if location 1777 contains 102070, it is interpreted as an ADCC# 0 0 instruction instead of an EJMP instruction.

b. Only a NOVA instruction may be placed in the last word of the file, because FEDIT may not increase the size of a file.

c. The ECLIPSE MV 32-bit Instruction Set, the ECLIPSE EDIT instruction operations, the ECLIPSE Vector (VCT) and Load Effective Address (LEF) instructions, and the NOVA Floating Point Instruction Set are not supported. The ECLIPSE Floating Point Instruction Set, ECLIPSE Standard Instruction Set, and NOVA Standard Instruction Set are supported.

d. ALC instructions have the following syntax:
mnemonic<carry><shift><#> acs acd

e. Memory reference instructions have the following format:
mnemonic <@>displacement <index>

   If *index* is omitted, then *displacement* is treated as an address.

f. Memory accumulator instructions have the following format:
mnemonic ac <@>displacement <index>

   If *index* is omitted, then *displacement* is treated as an address.

3. *ASCII format* displays and inputs data as one or two 8-bit pieces of data (bytes). Each byte can be entered as either an ASCII character, or as a numeric value not exceeding 377 (octal). To enter a numeric value, enclose the number in angle brackets (e.g., <32>). Valid examples of ASCII format are:

AB	Left byte = "A"	-	right byte = "B"
A<16>	Left byte = "A"	-	right byte = "16"
<74>x	Left byte = "<"	-	right byte = "x"
<16><33>	Left byte = "16"	-	right byte = "33"
v	Left byte = "0"	-	right byte = "v"
<14>	Left byte = "0"	-	right byte = "14"
A<0>	Left byte ="A"	-	right byte = "0"
61	Left byte = "6"	-	right byte = "1"

When the file is closed, a new checksum is calculated for the file so that subsequent reads of the file will not return file checksum errors. The message *filename.filetype* **UPDATED** verifies that the file has been closed. The program then returns to ADES-CLI.

Chapter 9
The Disk Edit Utility

The Disk Edit Utility (DEDIT.PRG) lets you display and patch any write-enabled disk in the system. DEDIT works on one disk sector (256 words) at a time.

Sectors are addressed by their block number. Block numbers are assigned sequentially, starting at cylinder 0, head 0, sector 0. For example, if a disk had 10 sectors/track and 2 tracks/cylinder, the sector at cylinder 4, surface 1, sector 3 would be (4 * 2 * 10) + (1 * 10) + 3 = 93.

When DEDIT is executed, you must first tell the utility which disk to edit. Remember that the disk you specify must be listed in the current System Equipment Table as a write-enabled media (e.g., SCRATCH or ADES). The following two questions are asked:

```
Device code?
Unit number?
```

Both the device code and unit number are entered in the current radix. If the device you specify is not listed in the current System Equipment Table as a write-enabled disk, you get either an EQUIPMENT NOT FOUND, WRITE PROTECTED MEDIA, or NOT A DISK, error message and the questions are repeated. If you wish to return to ADES-CLI to modify the current System Equipment Table, press the ESC key in response to one of the questions.

Once a write-enabled disk is specified, you enter the main loop:

```
Block number:    High [bnh]?
                 Low [bnl]?

Word address?

aaaaa xxxxxx yyyyyy<cr>
aaaaa xxxxxx yyyyyy<cr>
            :
            :
aaaaa xxxxxx yyyyyy<nl>
```

bnh is the default block number high, *bnl* is the default block number low. *aaaaa* is the current word address you are examining. *xxxxxx* is the value at the current word address. If you want to change the value, type in the new value (shown here as *yyyyyy*); otherwise, just type a delimiter. The delimiter you type controls what DEDIT does next. The following table summarizes the action each delimiter has.

Table 9-1. DEDIT Delimiter Actions

Prompt	Delimiter	Action
old value	<cr>	Close current location, open next location
	<nl>	Close current location, return to Word Address question
	<esc>	Do not modify current location, return to Word Address question
Word Address	<cr>	Open the location specified
	<nl>	Open the location specified
	<esc>	Close the currently opened block, and return to the Block Number question
Block Number	<cr>	Open the block specified
	<nl>	Open the block specified
	<esc>	Return to ADES CLI

WARNING: Using the ^C^A or ^C^B abort sequence causes the currently opened block NOT to be modified.

The block number, word address, and old and new values are displayed and entered in the current radix. The block number is treated as two 16-bit values (high and low order).

When a block is closed, the disk sector is updated and the message BLOCK XXX,YYY UPDATED is printed out (XXX is the high order block number and YYY is the low order block number).

Chapter 10
The File Display Utility

The ADES File Display Utility (DISPLAY.PRG) lets you display the contents of a file in both numeric and ASCII format. The numeric form is printed in the current radix.

When DISPLAY is executed, you must tell the utility which file to display:

```
Pathname?
```

Both the file's name and type (e.g., MYFILE.UDF) must be entered. If there is any problem opening the file, the system issues an error and repeats the question. If you wish to return to ADES-CLI at this point, press ESC.

DISPLAY then reads in and displays the contents of the file in 4 KW segments. Each line of the display represents 8 words of the file, listed in order from left to right. Column 1 is the address of the first word listed on the line. Columns 2-9 contain the numeric contents of the 8 words, and column 10 (the rightmost column) is the 16 ASCII character equivalent of the 8 words listed in columns 2-9.

Example:

```
0       000000 000000 000000 000000 000000 000000 000000 000000 ................
****
200     006217 000177 000000 000000 000000 000000 000000 000000 ................
210     000000 000000 000000 000000 000000 000000 000000 000000 ................
****
400     006203 000003 000671 030204 025102 020224 123400 106700 ......0.*B ....
410     006203 000016 006203 000005 000056 105000 006203 000016 ................
420     024226 006203 000004 006203 000004 031066 021017 101014 (.........26''..
                   :

                   :
1270    030204 041060 006213 005011 004501 042105 051440 043151 0.BO.....ADES F
1300    066145 020105 062151 072157 071040 020040 020040 051145 le Editor      R
                   :

                   :

                   :
```

"*****" in the left margin means that the locations not displayed at that point all contain zeroes. Bytes containing values less than 40 octal, or RUBOUT characters (177 octal), are displayed as "." in the ASCII format column.

Chapter 11
DTR Form Print Utility

The DTR Form Print Utility (DTRFORMS.PRG) lets you print one or more DTR forms at one time on the system console device. If a secondary output device is defined prior to calling the utility, the DTR forms will be printed on the secondary output device.

To execute the utility, type:

ADES-CLI> **DTRFORMS** *x*

where *x* is the number of forms you want to print.

When the forms are finished printing, the message DTR FORMS PRINTED is displayed.

Chapter 12
Contiguous Memory Test Utility (CMEMT)

The Contiguous Memory Test program (CMEMT.PRG) verifies that each 1 KW in memory exists. The tested memory range is from location 0 to the top of physical memory.

Top of physical memory is derived from the contents of system variable PMSZH?, located in Pseudo Page 0. The value of PMSZH? corresponds to the value of Word 2 in the System Equipment Table. Both values are defined at System Initialization, and are modified when the Manual Sizer's "M" operation is executed, or when a new System Equipment Table is read in via the **EQUIPMENT READ** ADES CLI command.

If the Operator Flag is ON, a banner message is printed; if no non-existent memory is detected (i.e., if memory is contiguous), a verification message is printed. If the Operator Flag is OFF, the only messages printed are those describing any detected non-existent memory.

The KW numbers printed refer to the address of that KW, and are printed in decimal. For example, in a 128 KW system, the first KW is 0, and the last KW is 127.

No SWREG bits are used by CMEMT.

Appendixes

Appendix A Glossary

Alignment programs: ADES test programs that help align the heads of magnetic recording devices.

Auto Sizer (ASIZE): program that determines the system configuration and places the information in the System Equipment Table.

Bootstrapping: the process of loading ADES from tape or disk into memory.

Checksum value: verifies the integrity of the memory-resident operating system code by summing all the memory-resident code and comparing the total to a known value.

CONLOG RSYS: on disk media, the name of the disk file in which the console log is saved.

Diagnostic programs: ADES test programs that detect and isolate hard faults on subsystems and modules. They also help to isolate a problem to the failing circuit by providing functions such as looping on an error.

Double keystrokes: a simple way to complement a flag in the Environment Control Word. When you have a prompt or when a program is awaiting input, you use CTRL-P plus the appropriate single keystroke command associated with the flag you wish to change.

DRVR file: driver file that contains executable code that calculates and performs I/O operations to peripherals supported by ADES.

Edit utility: serves as a text handling and a patch utility. Lets you edit any type of existing file and create source, text, or toggle files.

Environment Control Word: a word in memory that lets you control many aspects of your operating environment.

EQUIP0.UDF: a disk file to which you can save the information from the System Equipment Table.

Exerciser programs: ADES test programs which detect and isolate hard, intermittent, and interactive faults in systems and distributed systems. Their error reports indicate either which subsystem may be failing, or which FRU.

Formatter programs: ADES test programs which initialize magnetic media.

Initialization Monitor: program that takes over when you load ADES to initialize some system variables.

Kernel: that portion of the memory-resident system code that enables ADES to perform all of its system functions and utilities.

Logging: the process by which system console output is saved either in the disk file CONLOG.RSYS or on a log tape.

Macro: a file that contains a sequence of ADES CLI commands that execute automatically when you type in the name of the file; also referred to as a macro or SRCE (source) file.

Manual Sizer (MSIZE): a program that lets you add or modify information manually to the System Equipment Table about a system configuration.

Octal Debugging Tool (ODT): provides user with limited debugging facilities without using any of the ADES system routines. Uses one self-deleting breakpoint.

OLF file: an overlay file, which contains executable code.

PRG file: a program that runs under ADES; started at location 200 (8).

Radix indicator: changes the radix of an individual number. Add K for octal, H for hexadecimal, or . for decimal.

RSYS file: a system program that remains memory-resident once it is invoked.

Runtime media: the tape or disk containing ADES from which you are currently executing.

Screen Edit: a utility used when running on a video console that allows you to make changes to a command line before you execute it and/or to use the previous command line in forming the next one.

Script Builder: a program that lets you produce a script file to be saved on runtime media, or a one-page script that is executed, then discarded.

Script: a file that contains a sequence of ADES CLI commands that execute automatically when you type in the name of the file; also referred to as a macro or SRCE (source) file.

Single keystrokes: single digits or alpha characters used while a program is running to complement the meaning of any flag in the Environment Control Word.

SNRU file: a System Nonresident Utility file; it contains executable code (the code that is executed when an ADES system call is made).

SOP file: standalone program that is executed by typing in the program name in response to the FILENAME [ADES]? bootstrapping question. Standalone programs do not use any of the ADES system utilities. SOP files are started indirect through location 2.

SRCE file: a file that contains a sequence of ADES CLI commands that execute automatically when you type in the name of the file; also referred to as a macro or script.

SRCH file: a SEARCH test file (System Exerciser and Reliability Check) that is 2 KW in length and is written contiguously on disk media.

String: a temporary storage place that holds up to 80 characters; used to pass textual information to a program.

Switch register: a 32-bit register used for passing numerical information to ADES programs.

Symbolic Debugger (ADEB): lets you debug programs by accessing and changing the contents of any internal register or memory location. Uses 8 non-self-deleting breakpoints.

System Equipment Table (SET): a table in memory which lists all the peripherals and options in the system.

System identification: an 80-character string that describes software contained on the system media. Use the SYSID command to display or set up a message.

System Initialization Program (SINIT): takes over (if the Initialization Monitor encounters no problems) and concludes the power-up sequence.

System media: a tape or disk containing ADES.

System panic: a fatal ADES error.

Templates: characters (* and +) that serve as a shortcut in referring to filenames.

Timing programs: ADES test programs which provide help in calibrating equipment to meet timing specifications.

TOGL file: an operator-generated file, created through the Edit utility, that performs a specific function.

TXT file: an ASCII text file used for any purpose.

Update media: disk or tape media used to update runtime media, which must be disk. This media can be built using AOS **COPY** or RDOS **XFER** commands.

Update utility: lets you update an ADES system disk without rebuilding it.

Verification programs: ADES test programs which prove that a product performs according to its functional specification.

Write-protected media: a disk or tape that cannot be written to.

Appendix B
Power-Up Trouble-Shooting

If the bootstrapping process is successful, and the system hardware passes the test given by the ADES bootstrap program, the message FILENAME [ADES]? appears. If the test fails, the CPU is halted before the words are completely displayed. Use this appendix as a guide to help you discover what could be wrong. Table B.1 gives possible causes for certain messages you might get.

Table B-1. Power-Up Message Summary

Message	**Causes**	**Information Available**
Nothing	Wrong media Boot device failure Faulty console device Memory failure CPU failure	None
FILE or FILENAME	Memory failure	Information printed: GOOD, BAD, ADDRESS

Refer to Table B.2 for a listing of power-up tasks and their associated sequence numbers.

Table B-2. Power-Up Task Summary for Disk Systems

Sequence	Task Initiated
1	NOVA instruction test
2	Print "<cr><nl><nl>F"
3	Test memory locations 400-3777
4	Print "I"
5	Load Multisector Loader (MSL)
6	Print "L"
7	Checksum MSL
8	Print "E"
9	Size the system console
10	Test memory locations 0-377
11	Print "NAME"
12	Test memory locations 4000-top of memory
13	Print " [ADES]? "

NOTE: On tape systems, the message "<cr><nl><nl>FILE" is printed out after step 9, and steps 2, 4, 6, and 8 are not performed.

Errors that occur after you answer the **FILENAME** question will result in one of three error messages. The comments that follow the error message explain what could have caused it, and possible solutions.

1. FILE DOES NOT EXIST

   If you selected ADES by typing NEW LINE, then the boot program could not find either the Kernel program or the runtime media driver. If you selected a standalone program, the file you specified does not exist on the media.

2. FILE CHECKSUM ERROR

   The file just loaded did not checksum correctly. This could be because of bad media or memory problems. Try again. If the problem does not go away, then rebuild the media. If it still persists, then try another media. If that still does not solve the problem, then the cause is probably system-related (such as memory, drive, or controller).

3. DEVICE STATUS ERROR, DIA= *xxxxxx*

   The boot program driver received a DIA status error. *xxxxxx* is the octal value of the DIA register received.

Appendix C
Panic Codes

An ADES panic occurs when an unrecoverable error is detected. The following message appears:

FATAL ADES ERROR # *xxxxxxx*

xxxxxxx is the octal panic code. After the panic message, control is transferred to the ODT. The contents of the accumulators and carry are printed, and an ODT prompt is issued. (Refer to Table C.1 for a listing of panic codes and their meanings.)

To continue after a panic has occurred, use the ODT's **P** command. Depending on the severity of the error, the panic recovery routine may or may not work. If it does, you will get an ** ABORT ** message, followed by the ADES-CLI prompt. If the system doesn't recover, then manually reboot the runtime media.

On panics where a file was unable to be read in, AC0 contains the error code that describes the reason for read failure. Use the **ERMES** command for error code explanations.

Table C-1. Panic Codes (continued)

Panic Code	Description
0	Jump to 0.
1	Stack underflow.
2	Stack overflow.
3	Unable to access the teletype input package.
4	Fatal error return from Console Log Driver (CONLOG?). AC0 = error code.
5	Unable to access the CLI utility (file CLI.SNRU).
6	Unable to find filename of standard driver for system console.
7	Unable to read in system console driver file.
10	Unable to access CLIERR. AC0 = error code.
11	Attempt to output a character with system console driver undefined.
12	Unable to access abort module during an abort.
13	Kernel checksum error during 4 KW to 8 KW Kernel restoration.
14	System console or secondary output device timeout.
15	Unable to read in System Initialization program (SINIT).
16	Unable to read in PLB while trying to abort or terminate a script.
17	Unable to write out PLB while trying to abort or terminate a script.
20	SCALL? 0 attempted.
21	I/O error occurred before the system console device driver could be put into place. AS0 = device DIA status. AC1 = IOCB status (described below table).
22	Unknown model number received from ECLID instruction during CPU sizing.
23	Unable to load the CPU/Console/Memory sizer program CCMSZ.
24	Fatal error encountered within ERRMSG utility. AC0 = error code.
25	Unable to access EXCCHR within system console driver. AC0 = error code.
26	Duplicate filename found in directory structure.
27	Unable to access CLI support routines (file CLISR).

Table C-1. Panic Codes (concluded)

Panic Code	Description
30	Kernel checksum failure during an abort.
31	Unable to read in the runtime media's Master Disk Directory.
32	Unable to access control characters routine (file EXCCHR).
33	Unable to relocate RSYS back to the top of logical memory.
34	DIA status error received during a memory dump.
35	Directory segment integrity lost.
36	Device error from system console, or its slave processor or controller.
37	Unable to release file's memory resources after a file read error.
40	Unable to access terminate routines (file TERMINATE).
41	Unable to access Screen Edit utility (file SEDIT).
42	Jump indirect through location 1. (Interrupts illegally enabled.)
43	Unable to resolve KERNEL.RSYS during 4 KW to 8 KW Kernel restoration.
44	Nonresident system call made when caller was not at stack base level during 4 KW mode.
45	Unable to find secondary output device driver filename during 4 KW to 8 KW Kernel restoration.
46	Unable to read in secondary output device driver file during 4 KW to 8 KW Kernel restoration.
47	Unable to resolve runtime media driver during 4 KW to 8 KW Kernel restoration.
50	Runtime media driver checksum error during 4 KW to 8 KW Kernel restoration.
51	Rsys overwrite attempt while in 4 KW mode.
52	MPT mini-diskette read or write operation attempted with MPT mini-diskette 512-word buffer undefined.
53	Unable to read in Kernel program during a media switch.
54	Unable to initialize runtime media after an IORST.
55	Attempt to execute a 32-bit system call, utility, or initialization routine on a 16-bit processor.
56	32-bit instruction or extended memory test failed.
57	DOIO call executed with AC3 as source or destination accumulator.
60	Unable to access "GTSF.SNRU". AC0=error code.

The IOCB (I/O Command Block) status word is formatted as follows:

Bit 0	1 = IOCB active
Bits 1-4	Not used
Bits 5-7	Error Number
	0 = DIA status error (DIC for fixed head disks) 1 = DIB status error 2 = BMC error (BMC status cleared) 3 = controller full timeout 4 = operation timeout 5 = invalid model number 6 = invalid command
Bits 8-13	Not used
Bits 14-15	1 = done 3 = done + error

Appendix D
Mnemonics Descriptions

NOTE: Refer to the mnemonics HELP files for the latest information. To do so, type: **HELP MNEMONICS**.

Table D-1. Mnemonic Codes and Descriptions (continues)

Mnemonic	Model #	Description
ADCV*	4120	A/D Converter
	4130	A/D Converter
	4140	A/D Converter
	4150	A/D Converter
	4223	A/D Converter (Micro Products)
ALM	4255	ALM/4,8,16
	4227	Async Interface (Micro Products)
AP*	8620	Array Processor (AP130)
	8644	Array Processor (S/250)
	8661	Array Processor (IOP)
ATP	0	Attached Processor (DG10)
BMC	8642	High Speed Channel (S/250, C/350, M600)
	8699	High Speed Channel (S/140)
	8734	High Speed Channel (S/20)
	8772	High Speed Channel (S/280)
BMCT	9642	BMC Tester
BPC*	4348	Bit Synchronous Controller, 1 line
	4349	Bit Synchronous Controller, 4 lines
CDR	4016	Card Reader
CHAR	8614	Character Instruction Set (S/130)
	8639	Character Instruction Set (S/250)
	8664	Character Instruction Set (S/140)
	8731	Character Instruction Set (S/120)
COMM	8633	Commercial Instruction Set
CRC*	4228	CRC Gen/Chk (Micro Products)
	4266	CRC Gen/Chk

Table D-1. Mnemonic Codes and Descriptions (continues)

Mnemonic	Model #	Description
DACV*	4180	D/A Converter
	4224	D/A Converter (Micro Products)
	4300	DG/DAC
DCU	4250	Data Control Unit DCU/50
	4254	Data Control Unit DCU/200
DIF*	4335	Digital Interface Card (Micro Products)
DIO*	4065	Digital I/O
	4222	Digital I/O (Micro Products)
DKB	6063	1 MB Fixed Head Disk
	6064	2 MB Fixed Head Disk
DKM	8323	358 KB Mini-diskette (MPT series) 2.5 MB Diablo Disk
DKP	4047	6 MB Century Disk
	4048	25 MB Century Disk
	4057	92 MB CDC Disk
	4231	315 KB Floppy
	6030	10 MB Phoenix Disk
	6045	20 MB Gemini Disk
	6070	1.2 MB Quad Density Floppy
	6097	12.5 MB Echo Disk
	6099	25 MB Echo Disk
	6103	5 MB Cactus Disk
	6225	15 MB Cactus Disk
	6227	50 MB Daisy Disk
	6234	315 KB Floppy (Micro Products)
	6038	10 MB Phoenix Disk (Micro Products)
	6095	
DKT	6038	315 KB Floppy (Micro Products)
DPB	6224	15 MB BMC Cactus Disk (Micro Products)
	6280	50 MB BMC Daisy Disk (Micro Products)
DPF	6060	96 MB Zebra Disk
	6061	190 MB Zebra Disk
	6067	50 MB Zebra Disk
	6122	277 MB Zebra Disk
	6160	73 MB Kismet Disk
	6161	147 MB Kismet Disk
	6214	602 MB Kismet Disk
DPM	8632	Demand Paging Map
DPV	6236	360 MB BMC Argus Disk
DRTC	0	DCU Real Time Clock

Table D-1. Mnemonic Codes and Descriptions (continues)

Mnemonic	Model #	Description
DSK	4514	368 KB Mini-diskette
	6096	1.2 MB Quad density Floppy (Micro Products)
	6102	12.5 MB Echo Disk (Micro Products)
	6105	25 MB Echo Disk (Micro Products)
	6220	5 MB Cactus Disk (Micro Products)
	6222	15 MB Cactus Disk (Micro Products)
	6271	15 MB Harem Disk (DG)
	6267	368 KB Mini-diskette (DG10)
	6268	368 KB Mini-diskette (DG20,DG30)
ERCC	8400	Error Correction (S/230, C/330, S/200, C/300)
FPU	8612	Error Correction (S/130, AP130, C/150, S/250, C/350,
	8678	M600)
	8413	Error Correction (S/140) Floating Point (EAU)
	8539	Floating Point (Nova 3)
	8613	Floating Point (S/130)
	8622	Floating Point (C/150)
	8641	Floating Point (CPU 3,4)
	8662	Floating Point (S/140 Firmware)
	8663	Floating Point (S/140 Hardware)
	8731	Floating Point (S/120)
GDE*	4436	Graphics Display Entry Tablet (DG)
	4437	Graphics Display Entry Mouse (DG)
HHC*	8564	MicroNOVA Hand-held Console
HOST	0	DCU Host Interface
HSC*	4229	MicroNOVA High Speed Channel
IAC	4357	Intelligent Asynchronous Controller, 8 lines
	4358	Intelligent Asynchronous Controller, 16 lines
IOP	8660	I/O Processor
IOPT	0	I/O Processor Timer
IOT	8000	I/O Tester
	8001	I/O Tester (Micro Products)
IPB*	4240	Inter Processor Buss
ISC	4380	Intelligent Synchronous Controller, 2 lines
LPT	4034	240-300LPM
	6043	TP1 Printer
	6086	LP2 Printer
	6088	LP2 DCH Printer
	4216	240-900LPM DCH
	4320	Letter Quality
	4323	Band Printer

Table D-1. Mnemonic Codes and Descriptions (continues)

Mnemonic	Model #	Description
MBC	5220	D280C Color Display
	6012	DGC Display
	6040	Hardcopy Dasher, Infoton Display
	6052	LCD, Upper case
	6053	LCD, Upper/Lower case
	6106	D100/D200 Display
	6130	D400/D450 Display
	6150	G300 Graphics Display
	6166	D410 Display Workstation
	6167	D460 Display Workstation
	6168	D210 Display Terminal
	6169	D211 Display Terminal
MCAT	4206	MCA Transmitter
MDV	8534	Multiply Divide Option
MLPT	8690	MPT Series PIO Lineprinter
MMPU	8394	Nova 4 MAP
	8412	Eclipse MAP (S/200, C/300)
MMPU1	8618	Eclipse MAP (Other than S/200, C/300, S/140)
	8678	Eclipse MAP (S/140)
MMU	8535	Nova 3 MAP (no protection)
MPT	8321	MPT/80, MPT/83, MPT/87 Console Interface
	8322	MPT/100 Console Interface
MPU	8538	Nova 3 MAP
MRTC	0 6020	Micronova RTC Standard Tape
MTA	6026	Dual Mode Tape
	6123	Slingshot (Micro Products)
	6125	Slingshot (Nova/Eclipse)
	6230	Oasis Cartridge Tape Subsystem
	6231	Oasis Cartridge Tape Subsystem (Micro Products)
	4307	Dual Mode GCR
NBA	4460	Network Bus Adapter (Micro Products)
NVM*	8316	Non-volatile Memory
PAR	8536	Parity Option
PIT	4217	Programmable Interval Timer
PLT*	4017	Plotter
	4435	Graphics Plotter (DG)
PROMB*	8574	Prom Burner (Micro Products)

Table D-1. Mnemonic Codes and Descriptions (continues)

Mnemonic	Model #	Description
PTP	4012	Paper Tape Punch
PTR*	6013	Paper Tape Read
QTY	4060	Quad Mux
RTC	4008	Real Time Clock
SDX*	8320	MBC/SDX I/O Board
SLM	4263	SLM/2
	4226	Sync Interface (Micro Products)
TLC*	4516	Talker/Listener Controller (Micro Products)
	4517	Talker/Listener Controller (NOVA/ECLIPSE)
TTI	5220	D280C Color Display
	6012	DGC Display
	6040	Hardcopy Dasher, Infoton Display
	6052	LCD, Upper case
	6053	LCD, Upper/Lower case
	6106	D100/D200 Display
	6130	D400/D450 Display
	6150	G300 Graphics Display
	6166	D410 Display Workstation
	6167	D460 Display Workstation
	6168	D210 Display Terminal
	6169	D211 Display Terminal
	6265	Color Graphics Display (DG)
TTO	4433	136-column, 160 CPS Dot Matrix Printer (DG)
	4434	80-column, 160 CPS Dot Matrix Printer (DG)
	4518	136-column, 35 CPS Letter Quality Printer (DG)
	5220	D280C Color Display
	6012	DGC Display
	6040	Hardcopy Dasher, Infoton Display
	6052	LCD, Upper case
	6053	LCD, Upper/Lower case
	6106	D100/D200 Display
	6130	D400/D450 Display
	6150	G300 Graphics Display
	6166	D410 Display Workstation
	6167	D460 Display Workstation
	6168	D210 Display Terminal
	6169	D211 Display Terminal
	6265	Color Graphics Display (DG)

Table D-1. Mnemonic Codes and Descriptions (concluded)

Mnemonic	Model #	Description
ULM	4243	ULM/5 Sync/Async Line Multiplexor
	*4336	Async/Sync Line Multiplexor (Micro Products)
	4342	ATI-16 Asynchronous Terminal Interface
	*4463	Pearl Asynchronous Line Multiplexor (DG)
UPSC	0	Universal Power Supply Controller
VID*	4337	Video Interface Board (Micro Products)
WCS*	8415	Writeable Control Store (S/200, S/230)
	8615	Writeable Control Store (S/130)
	8638	Writeable Control Store (S/250)

*Not autosized

Index

CUT ALONG DOTTED LINE

Corporate Publications Comment Form

Title: ____________________

Document No. ____________________

Please help us improve our future publications by answering the questions below. Use the space provided for your comments.

IS THIS MANUAL EASY TO USE?

Can you understand it?

Can you find things easily?

Is the language appropriate?

Are technical terms defined as needed?

IS THE MANUAL WELL WRITTEN?

Is it well organized?

ARE THE ILLUSTRATIONS HELPFUL?

Are the visuals well designed?

Are the labels and captions clear?

IS THE MANUAL HELPFUL?

Does it tell you all you need to know?

What additional information would you like?

Are any points belabored?

IS THE INFORMATION ACCURATE?

(If not, please specify with page number and paragraph.)

Name: ____________________

Dept: ____________________ MS: ____________________

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:

DataGeneral

Corporate Publications (C-138)
4400 Computer Drive
Westboro, MA 01581

[illegible]
Diagnostic
Executive
System

ADES

Operator's Manual

[illegible]

014-000[illegible]-03