XXDP User's Manual

Order Number AA–FK83A–TC

This manual supersedes the *XXDP User's Manual*, AZ–GMJAA–MC.

digital equipment corporation
maynard, massachusetts

First Printing, May 1985
Revised, May 1989

COMPACTape	LGP	RT
DEC	MASSBUS	ThinWire
DECmate	MicroPDP-11	UNIBUS
DECnet	MicroVAX	VAX
DECUS	PDP	VAXBI
DECwriter	P/OS	VAXcluster
DELNI	Professional	VAXstation
DELQA	Q22-bus	VMS
DEQNA	Rainbow	VT
DESTA	RSTS	Work Processor
DIBOL	RSX	digital™

S1130

This document was prepared using VAX DOCUMENT, Version 1.1.

Contents

Chapter 4 UPDAT Utility

Chapter 7 XTECO Utility

Chapter 8 DXCL (DEC/X11 System Exerciser Utility)

Tables

Preface

Intended Audience

This manual is intended for field service personnel who use the XXDP operating system to load and run diagnostics of Digital's PDP–11 and MicroPDP–11 families of processors and their associated peripherals.

Document Structure

This manual contains both introductory and reference information and is divided into nine chapters, six appendixes, and a glossary. Each chapter is divided into a descriptive section that introduces basic concepts and a command section that describes the associated commands in alphabetical order. Each chapter has several examples of actual usage.

- Chapter 1 introduces the XXDP diagnostic operating system, its components, services, utility programs, device drivers, and terminal interface and file conventions.
- Chapter 2 discusses the XXDP monitors—the SM monitor and the XM monitor, including how to start them up and communicate with them.
- Chapter 3 discusses XXDP's diagnostic runtime services (DRS) and how to use them to customize your diagnostic environment.
- Chapter 4 discusses the UPDAT utility, a file manipulation program used for building XXDP media and handling files residing on them.
- Chapter 5 discusses the PATCH utility, used to modify binary formatted files stored on an XXDP media.
- Chapter 6 discusses the SETUP utility, used to build a diagnostic's hardware and software tables.
- Chapter 7 discusses the XTECO utility, used to create and edit text files, such as XXDP batch control files.
- Chapter 8 gives an overview of the DXCL (DEC/X11 runtime exerciser) utility.

- Chapter 9 discusses the batch control facility, used to run diagnostic files without operator intervention.

Eight appendixes provide reference information.

- Appendix A provides a command summary.
- Appendix B lists devices supported.
- Appendix C lists component names.
- Appendix D discusses building XXDP.
- Appendix E provides user tips to facilitate XXDP operation.
- Appendix F lists DEC/X11 device/option modules and software switch options and provides a sample of an RTE build.
- Appendix G lists and explains XXDP error messages.
- Appendix H lists and explains RTE configurator/linker messages.
- The Glossary provides a dictionary of XXDP-related terms.

Conventions

This book uses the following conventions:

;	Semicolons within code examples or examples of user-system interaction introduce comments on the code or interaction.
[ ]	Brackets enclose optional fields in command arguments.
$\boxed{\text{Ctrl/key}}$	Control keys are shown as $\boxed{\text{Ctrl/key}}$. For example, Control C is shown as $\boxed{\text{Ctrl/C}}$.
$\boxed{\text{Key}}$	Words outlined by a box indicate a key on your keyboard.
lowercase characters	Lowercase characters in computer commands indicate input that you replace with an appropriate word or phrase. Optional characters in a command are also shown in lowercase.
<u>Underlined characters</u>	Underlined characters in examples of user-system dialogue indicate what the system prints/displays on the console terminal.
UPPERCASE CHARACTERS	Uppercase characters in computer commands indicate input that you must type exactly as shown.

Associated Documents

The following is a list of related documents:

- *DEC/X11 User's Manual*
- *XXDP DEC/X11 Quick Reference Guide*
- *XXDPV2 Driver Programmer's Guide*
- *XXDPV2 File Structure*

Chapter 1
Introduction

XXDP is the diagnostic operating system for the PDP/LSI-11 family of systems.

1.1 Components

The XXDP system consists of the following major components:

- Monitor
- Diagnostic runtime services
- Utility programs
- Loadable device drivers

These four components work together to accomplish the system functionality.

1.1.1 Monitor

The monitor is the highest level software and forms the core of the system. All other components require monitor support for their operation. The monitor program provides:

- Load and execution of diagnostics
- Console terminal services
- Batch control
- File services for the system media

The system media is the storage media on the device from which the monitor is loaded. All other components use the terminal services for operator communications and the file services for accessing files on the system media.

Chapter 2 describes the monitor and its commands.

1.1.2 Diagnostic Runtime Services

The diagnostic runtime services (DRS) are an extension of the monitor; they provide nondiagnostic functions for certain types of diagnostic programs, commonly referred to as supervisor or DRS-compatible diagnostics.

Among the functions that the DRS provide are:

- Standard operator interface
- Error message formatting
- Control of diagnostic operation

Chapter 3 describes the DRS and their commands.

1.1.3 Utility Programs

The utility programs perform various file operations. They use the monitor for printing/displaying and receiving messages and for loading read/write device drivers required for file operations.

The following utility programs are available:

- UPDAT–Chapter 4 describes UPDAT and its commands.
- PATCH–Chapter 5 describes PATCH and its commands.
- SETUP–Chapter 6 describes SETUP and its commands.
- XTECO–Chapter 7 describes XTECO and its commands.
- DXCL–Chapter 8 describes DXCL and its commands.

1.2 Loadable Device Drivers

The loadable device drivers are the device handlers used by the various utility programs to access storage media and I/O devices. The drivers are resident on the system storage media and loaded into memory as required.

1.3 XXDP Conventions

You use two kinds of conventions to communicate with XXDP: terminal interface conventions and file conventions. Section 1.3.1 and Section 1.3.2 describe these two kinds of conventions.

1.3.1 Terminal Interface Conventions

The XXDP monitor supports the console terminal. The console terminal driver is a simple, flag-driven handler.[1]

The driver makes no distinction between uppercase and lowercase characters and supports all printing characters. However, some characters have special meaning in commands, arguments, or text files. Table 1–1 lists these characters and their function. The remaining chapters give descriptions and examples of their use.

Table 1–1: Special XXDP Characters

Character	Use
:	Delimit a device specification (e.g., DU0:)
;	Start comment line in a batch control file
.	Indicate beginning of a file name's 3-character extension
?	Substitute a single-character (wildcard) in a file name
*	Substitute multiple characters (wildcard) in a file name
=	Separate input and output specifications in a command
<	Provide equivalent of =
/	Indicate command switch

The terminal driver supports the control characters listed in Table 1–2. You enter control characters by simultaneously pressing the Ctrl key and the designated letter.

Table 1–2: XXDP Control Characters

Character	Use
Ctrl/C	Stop current activity and return control to program (testable in batch control file)
Ctrl/Z	Stop current activity and return control to program (not testable in batch control file)
Ctrl/U	Delete entire line of input so new line can be entered

[1] Flag-driven means that no interrupts are used and thus unsolicited interrupts do not interfere with diagnostic programs.

Table 1–2 (Cont.): XXDP Control Characters

Character	Use
Ctrl/S	Inhibit typing (XOFF)
Ctrl/Q	Resume typing (XON)
Ctrl/X	Resume activity after WAIT command in batch control file

You specify XXDP files by a name and an extension. The format of a file specification is **filnam[.ext]**.

A file name can be from one to six characters in length, and the extension (**ext**) can be from one to three characters in length. A dot (.) separates the name from the extension. A file name can consist only of alphanumeric characters (A–Z and 0–9) and cannot contain embedded spaces.

Sample file names are:

UPDAT.BIN

DRSSM.SYS

TEST.1

XMONC0.LIB

You can use the wildcard characters asterisk (*) and question mark (?) in file specifications. Use the ? as a substitute for a single character and the * as a substitute for a string of characters.

For example, if you want a list of all the files with the .CCC extension on a media, you can give the specification *.CCC.

Other examples of the wildcard characters are:

XMON??.LIB	All files whose name starts with the characters XMON and ends with any two other valid characters (or nulls) and have the extension .LIB
XMON*.LIB	Same effect as above
XMONC0.*	All files with the name XMONC0 and any extension
XMONC0.???	All files with the name XMONC0 and a 3-character extension

1.3.2 File Extensions

Some extensions identify particular file types. For example, batch files have .CCC extensions. The following table lists extensions that have particular meanings.

Extension	Meaning
BIN	Executable program file that may not be run or loaded in batch control operation
BIC	Executable program file that may be run or loaded in batch control operation
SYS	System file
CCC	Batch control file
OBJ	DEC/X11 object module
LIB	Library file
TXT	Text file
BAK	XTECO backup file

Chapter 2

Monitors

The XXDP V2 operating system consists of two monitors: the SM (or small) monitor and the XM (or extended) monitor. The XM also contains a small monitor (SM) emulator. You use the XM monitor when a memory management unit (MMU) exists on the system and passes the requirements for booting this monitor. You boot the SM monitor by issuing a batch file request. You use the SM monitor in cases when the XM monitor cannot be booted. You can also emulate the SM when XXDP is running under the XM. The SM monitor emulates the capabilities of the XXDP V1 monitor, with some restrictions.

This chapter describes the monitor and lists the commands for each of its two categories.

2.1 Description

The V2 monitor is simple to boot, configure and use. The monitor is relocatable and thus supports the loading of current diagnostic products. It self-starts if all but a 1.5-Kword reserved section is corrupted during diagnostic operations. The monitor does not rely on hardware interrupts for its operation, thereby minimizing dependency on hardware functionality and impact of malfunctioning equipment.

2.1.1 Monitor Size and Components

At load time the XXDP monitor is about 8 Kwords in size and consists of three major sections: secondary bootstrap, initialization code, and runtime monitor code. The secondary bootstrap is loaded into memory at boot time and in turn loads the remainder of the monitor into memory. The initialization code gathers certain system information and relocates the runtime monitor. The runtime monitor is the code that is used to carry out the various operator functions. Section 2.1.6 describes the start-up process in detail.

The runtime monitor consists of five sections:

- Read-write device driver—loads programs from the system media and reads batch control files
- Console terminal driver

- Monitor services handler—processes requests for monitor services that are made by utility programs via the EMT (trap emulator) instruction
- Operator interface handler—processes operator commands from the console terminal
- Batch control handler—processes batch files from the system media

The runtime monitor is approximately 2 Kwords in size. Since older diagnostic programs expect the monitor size to be 1.5 Kwords, the monitor's lower .5 Kwords may be overwritten by a diagnostic program and then later restored by the monitor.

2.1.2 Diagnostic Requirements

Memory and CPU diagnostics that are compatible with XXDP V1 are compatible with XXDP V2. They must maintain the integrity of the 1.5-Kword area at the top of the first 28 Kwords of memory. XXDP XM autoboots if the monitor is destroyed, but all conditions may not be fully restored.

The following is a list of CPUs presently known to have the MMU support required to be supportable by the XXDP XM monitor:

- PDP-11/23 family
- Micro PDP-11
- PDP-11/53, 11/73, 11/83
- Unibus–PDP-11/24, PDP-11/34, PDP-11/35, PDP-11/40, PDP-11/44, PDP-11/45, PDP-11/70, PDP-11/84

2.1.3 Hardware Requirements

Memory management is required for the XXDP XM monitor to operate. If there is a failure in the MMU, the system reports the error and boots the XXDP SM monitor.

The system reports additional information helpful in determining the cause of the problem. However, the XXDP operating system does not perform rigorous error detection and reporting, because it is not intended to be a diagnostic.

2.1.4 Diagnostic Restrictions

Diagnostic restrictions and their explanations are as follows:

- **Restriction**: You cannot use many of the SYSMAC diagnostics.

  Explanation: Diagnostics that interfere with the monitor EMT (trap emulator) and TRAP vector area cannot use system calls. Many SYSMAC diagnostics use these areas when they load and run.

- **Restriction**: The XXDP SM monitor requires a minimum of 16 Kwords of memory.

  Explanation: The boot section requires 8 Kwords of memory plus room for the monitor. Diagnostics that require the PDP-11 Diagnostic Supervisor also require this minimum environment.

- **Restriction**: The integrity of the 1.5-Kword base monitor or root at the top of the first 28 Kwords of memory must be maintained. Failure to do so will require a manual reboot for continued system operations.

  Explanation: This root required to restore the monitor.

2.1.5 Differences Between XXDP V2.4 and Previous Versions

The current version of XXDP (V2.4) differs from previously released versions of XXDP V2 in the way it runs programs.

- In former versions, programs were run under the same monitor whose name was displayed at boot time. That is, if you booted the XXDPSM, you ran your programs under the small monitor; if you booted the XXDP extended monitor, you ran your programs under the extended monitor. However, if you had a program that ran under one monitor and you had booted the other, you had to reboot to run that program.

- In V2.4 and above, you do not have to reboot. XXDP runs programs after checking location 52, bit 12. If it is set, XXDP runs the program under the extended monitor; if it is clear, the program runs under XXDPSM.

- Use the UPDAT or PATCH utility to set this bit. If you use UPDAT, you may have to reset the diagnostic's lower memory limit. Use the LOCORE command (see Section 4.2.17), making sure to issue the CLR command before loading the program and changing the limit. See Chapter 5 for information on using the PATCH utility.

  If you are unable to use the UPDAT or PATCH utility to set bit 12, refer to the XXDP HELP facility descriptions of the DEFSM, DEFXM, and CLEAR commands.

2.1.6 XXDP System Start-up Procedure

Follow these steps to start up the system:

1. Halt the processor (after making sure that operating software has been gracefully shut down).
2. Mount/load your XXDP media.

   CAUTION: *If you are working on a system that has unknown hardware problems, make sure that the load device is write-disabled.*
3. Reenable and boot the processor.

After the monitor has successfully loaded, it identifies itself and gives the drive number from which it was booted (multidrive devices only) and the memory size, as in the following example:

```
XXDP-SM SMALL MONITOR VERSION 2
BOOTED FROM DY0
24 KW OF MEMORY
NON-UNIBUS SYSTEM
```

The message in this example is printed after you boot the XXDP SM using drive 0 from an RX02 on a system with 24 Kwords of memory.

After these steps have been successfully completed, the monitor prints/displays a dot (.) to prompt you for commands. Section 2.2 discusses SM monitor commands; Section 2.3 discusses XM monitor commands.

2.1.7 Start-up Process

Here is what happens when you boot the load device.

The first physical block of data on the media is loaded into the first 256 (decimal) words of memory. This data is the secondary bootstrap. Control is passed to it from the hardware or primary bootstrap.

The secondary bootstrap reads the remainder of the monitor from the load media into memory. Once the load is complete, the secondary bootstrap passes control to the initialization code and the boot process is complete.

NOTE: *If a detectable error occurs during the secondary bootstrap operation, the processor halts.*

If the system has successfully carried out the above process, XXDP now performs these functions:

1. Reads and executes the BOOT.CCC batch file and sets up conditions from this file (either SM or QUIET [see Section 9.2.4])

2. Sizes memory (up to 124 Kwords) and sizes for the presence of standard line or programmable clocks (KW-11L and KW-11P), processor type, and interrupt integrity
3. Verifies hardcore requirements
4. Reports errors found
5. Informs the user if it is booting the SM monitor instead of the XM monitor (if not in QUIET mode)
6. Loads and locates the monitor to the top of memory and transfers control to this code. In the case of the SM, the monitor can relocate up to 28 Kwords

   If you boot the XM monitor, it can load additional code into extended memory. XM is normally loaded. If the XM monitor cannot be loaded or if the BOOT.CCC file instructs otherwise, the SM monitor is loaded.
7. Identifies the monitor if not in QUIET mode
8. Starts the monitor

The BOOT.CCC file is executable only by the boot section of code and accepts either SM (see Section 2.2) or QUIET (see Section 9.2.4) commands. The QUIET command puts the XXDP XM monitor into QUIET mode and starts the SYSTEM.CCC batch file. You can use only one of these commands.

The XXDP SM monitor starts if the system is too small for the XM monitor or if the MMU is not operating properly. Since the XXDP SM monitor is not capable of QUIET mode operation, the QUIET command is ignored by this monitor.

NOTE: *The XXDP monitor assumes that you are operating with U.S.-type (60 Hz) power. If you are using European-type (50 Hz) power, you must modify the monitor. Location 370 in XXDPXM.SYS contains an indicator or power type. Enter 0 for 60 Hz and non-zero for 50 Hz. Chapter 4 (on UPDAT) explains at length how to modify files.*

2.2 SM Monitor Commands

Table 2–1 lists XXDP SM commands. Section 2.2.1 through Section 2.2.7 discuss each command. You must type the first letter only of all SM commands except the VT command, which requires two characters.

Table 2–1: XXDP SM Monitor Commands

Command	Function
Chain	Run a batch job
Directory	List directory of load media
Enable	Enable alternative drive for system device
Help	Print/display help information
Load	Load a program
Run	Run a program
Start	Start a program
VT	Change the console terminal type

Some commands have optional switches, which consist of a single character preceded by a /. You use switches to modify the command function.

2.2.1 CHAIN Command

You use the CHAIN command to initiate execution of a batch (chain) file.

The format of the command is:

Chain filnam[/QV]

where:

- **filnam** is the name of the batch control file you want to run. The file must reside on the system media and have a .CCC extension.
- **/QV** is an optional switch directing that the diagnostics listed in the batch file be run for one pass only, even if diagnostics contained in a batch file call for more extensive testing.

Chapter 9 describes batch control in detail.

2.2.2 DIRECTORY Command

You use the DIRECTORY command to obtain a list of all the files on the system media.

The format of the command is:

Directory [/L],[/Fast]

where:

- **/L** is an optional switch that causes the directory to be printed on a line printer rather than the console terminal.
- **/Fast** is an optional switch that causes the directory to be printed in a short form, which consists of only the entry number and file name.

XXDP prints/displays either a long or a short directory.

- The long directory lists five items of information:
  1. Entry number
  2. File name and extension
  3. Date the file was created, followed by the letter "C," if the file is contiguous (that is, its blocks follow one another on the media). Most files are "linked"; that is, their blocks are not in order on the media.
  4. Length of the file in 256-word (decimal) blocks
  5. Number of the first block in the file
- The short directory lists only file names.

The directory utility (DIR.SYS) and the read/write device driver for the system media type must reside on the system media in order for the directory command to work. If one of these files is not on the media, the monitor prints/displays an error message.

Figure 2–1 illustrates an SM long directory. Figure 2–2 illustrates an SM short directory.

Figure 2–1: SM Directory Long Form

```
ENTRY#  FILNAM.EXT        DATE         LENGTH     START
     1  XXDPSM.SYS     02-JUN-87         12       000100
     2  DY    .SYS     02-JUN-87          5       000120
     3  DIR   .SYS     02-AUG-87          6       000066
```

Figure 2–2: SM Directory Short Form

```
1  XXDPSM.SYS
2  DY    .SYS
3  DIR   .SYS
```

2.2.3 ENABLE Command

NOTE: *The ENABLE command is valid only for multidrive devices and affects drives, not controllers. The devices must be of an identical type, for example, two devices of type DU: or two of type MU:.*

You use the ENABLE command to designate a different drive as the system (default) device without rebooting the system. For example, if you booted the system from drive 0 of an RX02 and now want the monitor to use drive 1 as the system (default) device, use this command to make the change without rebooting the monitor.

The format of the command is:

Enable drive_number

where:

- **drive_number** is the number (octal) of the drive you want to enable. Drive numbers are in the range of 0 to 7.

A valid XXDP V2 media must reside on the drive you specify or the system does not complete the command and prints/displays an invalid device error. (See Appendix G.)

The following is an example of using the ENABLE command:

```
.E 1
```

This command enables drive 1 as the system device.

2.2.4 HELP Command

You use the HELP command to obtain a brief summary of XXDP commands.

The format of the command is:

.Help

XXDP prints/displays the contents of a file named HELP.TXT. This file must reside on the system media.

The following is an example of using the HELP command:

```
.H
```

2.2.5 LOAD Command

You use the LOAD command to load a file into memory.

The format of the command is:

Load filnam[.ext]

where:

- **filnam** is the name of the file that you want to load.
- **[.ext]** is the optional extension of the file that you want to load. If you omit the extension, XXDP uses either .BIN or .BIC. If the media contains a file with the specified name and both extensions, XXDP uses the first file that it finds.

After you issue this command, XXDP prints/displays the full file name of the loaded program. The LOAD command loads a program into memory, but XXDP does not start the program. You can think of the LOAD command as the first half of a RUN command (see Section 2.2.6).

The following are examples of using the LOAD command:

```
❶  .L DIAG
❷  .L ZDJCA2.NEW
```

❶ Load DIAG.BI?

❷ Load ZDJCA2.NEW

2.2.6 RUN Command

NOTE: *The RUN command combines the LOAD and START commands. (See Section 2.2.5 and Section 2.2.7.)*

You use the RUN command to load and start a diagnostic program or utility that is stored on the load or system media.

The format of the command is:

Run filnam[.ext] [addr]

where:

- **filnam** is the name of the file you want to run.

  You can use wildcard characters in the file specification. The monitor loads and runs the first file found that fits the wildcard description.

- **ext** is the optional extension of the file that you want to load. If you omit the extension, XXDP runs either a .BIN or .BIC file. If the media contains a file with the specified name and both extensions, XXDP uses the first file that it finds.

- **addr** is the program's optional transfer address (octal). XXDP starts the file at 200 octal if you indicate no transfer address. You can specify a starting address if you want the file to start at a different point.

After XXDP finds and loads the program it prints/displays its full name so that you can verify the load before XXDP starts the program. You can use this report to check that the right program will be run, especially if you have specified a wildcard.

The following are examples of using the RUN command:

```
❶  .R UPDAT

❷  .R SAMPLE.XXX

❸  .R RXDIAG 204
```

❶ Load and start the UPDAT utility (UPDAT.BI?)

❷ Load and start SAMPLE.XXX

❸ Load and start RXDIAG.BI? at location 204

2.2.7 START Command

CAUTION: *Do not issue commands between the LOAD and START commands. You may overwrite the program you have loaded.*

You use the START command to start a file that you have already loaded into core memory using the LOAD command.

The format of the command is:

Start [addr]

where:

- **addr** is the program's optional transfer address (octal).

XXDP starts the file at 200 octal if you indicate no transfer address. You can specify a starting address, if you want the file to start at a different point.

The LOAD and START command sequence lets you load a program and halt the processor so that you can modify memory contents. You then restart the program.

The following are examples of using the START command:

```
❶ .L RXDIAG
❷ .S

❸ .L RXDIAG
❹ .S 204
```

❶ Load RXDIAG.BI?

❷ Start at transfer address

❸ Load RXDIAG.BI?

❹ Start at 204

2.2.8 VT Command

You use the VT command to change the console terminal type back and forth between video display terminal and hard-copy terminal.

The format of the command is:

.VT

When you boot the small monitor, the normal terminal type is hard-copy.

2.3 XM Monitor Commands

This section describes the XM monitor commands.[1]

Table 2–2 summarizes the XXDP XM monitor commands.

Table 2–2: XXDP XM Monitor Commands

Boot	Boot XXDP from the specified device
BOOTSM	Boot the small monitor
Chain	Run a batch job
CLEAR	Return the monitor to a neutral state (negates DEFSM and DEFXM)
COpy	Copy a file or device
DAte	Set the date or report the date
DEFSM	Set default operation to small monitor (by activating the SM emulator)
DEFXM	Set default operation to extended monitor (by deactivating the SM emulator)
DElete	Delete a file
Directory	List directory of load media
Enable	Enable alternative drive for system device
Help	Print/display help information
Initialize	Initialize a device
Load	Load a program
Print	Print a file on the system line printer
REname	Rename a file to a new name
Run	Run a program
SEt	Set device or system parameter
Start	Start a program
TEST	Enter user friendly mode by running SYSTEM.CCC
Type	Type a file
VErsion	Print out information about the extended monitor

[1] The XM monitor no longer supports the FILL command.

2.3.1 Abbreviating Keyboard Commands

You can use abbreviations to enter a command. Valid abbreviations are the minimum number of characters required to make the command or switch unique. In the following sections the required part of a command or switch is shown in uppercase and the optional part in lowercase.

2.3.2 XM Command Syntax

The XM monitor accepts a command in either of these forms:

- A complete string containing all the information necessary to execute a command
- A partial string, in which case the system prompts for the rest of the information

NOTE: *Only the monitor prompts you for input; XXDP utilities do not.*

General syntax for a command is one of the following:

command[/switch] input_filespec output_filespec

or

command[/switch]
<u>**prompt1?**</u> **input_filespec**
<u>**prompt2?**</u> **output_filespec**

where:

- **command** is the command name.
- **/switch** is a command qualifier.
- <u>**prompt**</u> is the monitor prompt that asks for more information. The monitor prints/displays an appropriate prompt, only if you leave out an input or output file or device specification.
- **input_filespec** is the name of the file on which the action is to be taken.
- **output_filespec** is the name of the file that is to receive the results of the action.

A **filespec** identifies a file, its extension, and the device on which it is stored. You can sometimes omit the device name, file name, or extension as directed, because XXDP assumes a specific value if you don't supply any. **Filespec** syntax is:

dev:filnam.ext

where:

- **dev:** is a device name.
- **filnam** is the 1- to 6-character name of file. Wildcard characters are permitted in the file specification.
- **.ext** is the 1- to 3-character file extension.

2.3.3 BOOT Command

You use the BOOT command to direct the monitor to boot another XXDP monitor from another XXDP device.

The format of the command is:

Boot dev:

where:

- **dev:** is the name of the device that you want to boot. The appropriate device handler must be present on the system device.

The following is an example of using the BOOT command:

```
.B DU0:
```

2.3.4 BOOTSM Command

You use the BOOTSM command to boot the small monitor when diagnostics are running under the extended monitor.

The format of the command is:

BOOTSM

XXDP prohibits the use of the extended monitor commands.

The small monitor (XXDPSM.SYS) must reside on the system media.

2.3.5 CHAIN Command

You use the CHAIN command to run a batch (chain) file.

The format of the command is:

Chain filenam[/QV]

where:

- **filnam** is the name of the batch file you want XXDP to run. The file must reside on the system media and have a .CCC extension.

- **/QV** is an optional switch directing that the diagnostics listed in the batch file be run for only one pass. If you omit this switch, the file runs indefinitely.

Chapter 9 describes batch control in detail.

2.3.6 CLEAR Command

You use the clear command to clear the XM or SM flag.

The format of the command is:

CLEAR

Once the flags are cleared, the monitor can check location 52, bit 12 before running a diagnostic program. If the bit is set, XXDP runs the program under the extended monitor; if it is clear, XXDP runs the program under the small monitor.

The XM and SM flags are cleared at boot time.

2.3.7 COPY Command

You use the COPY command for file transfer and maintenance operations.

The format of the command is:

COpy [switch-list] idev:[ifilnam.ext] odev:[ofilnam.ext]

where:

- **switch-list** is one or more switches. You can use the following optional switches in any combination:
  - **/Boot** copies the root monitor from the input device into the boot block of the output device. The command places the appropriate secondary bootstrap in the boot block and places the monitor file on the media in a predetermined section. The root monitor consists of XXDPSM.SYS merged with the driver of the output device.
  - **/DElete** automatically deletes existing file(s) of the same name residing on the output device before it copies input files.
  - **/DEVice** copies a device in image mode (a block-for-block transfer) to a like device. A copy to a device which is not identical causes the command to terminate without completing. Bad block devices may halt execution of the command without completing it.
  - **/Files** copies all files from the specified input device. If a file of the same name already exists on the output device, the input file is not copied.

— **/NORewind** is a switch that prevents XXDP from rewinding the tape to which files are being copied. If you omit this switch, XXDP rewinds the tape after each copy operation. Set the tape at the beginning-of-tape (<BOT>) before you use this switch.

- **idev:** is the optional name of the device where the file(s) you want to copy reside. If you omit the device name, the monitor copies from the system device.
- **ifilnam.ext** are the name and extension of the file that you want to copy.
- **odev:** is the optional name of the device where the file(s) you want to copy are placed. If you omit the device name, the monitor copies the new file to the input device.
- **ofilnam.ext** are the name and extension of the file that receives the copy. If you omit the file name, the system gives the output file the same name as the input file.

The following is an example of using the COPY command:

```
.COPY A.BIN DY0:
```

This command copies the file A.BIN from the system device to DY0: using the same name.

2.3.8 DATE Command

You use the DATE command to find or to set the system date.

The format of the command is:

DAte [dd-mmm-yy]

where:

- **dd** is the day of the month (a decimal number from 1 to 31).
- **mmm** is the first three characters of the name of month.
- **yy** is the year (a decimal number from 83 to 99).

If you give the command with a date, that date is set. If you give the command without a date, the current date is printed/displayed. If no date has been set, the system prints/displays the date of 1-JAN-83.

The following are examples of using the DATE command:

```
❶ .DATE 18-MAY-83

❷ .DATE
```

```
18-MAY-83
```

❶ Sets the date

❷ Shows you the current date

2.3.9 DEFSM Command

You use the DEFSM command to set the SM flag in order to run diagnostics designed for the SM monitor.

The format of the command is:

DEFSM

This command sets the SM flag and disables the checking of location 52, bit 12. DRS-compatible programs now run by default under XXDP's small monitor emulator, until you issue the CLEAR command (see Section 2.3.6).

2.3.10 DEFXM Command

You use the DEFSM command to set the XM flag in order to run diagnostics designed for the XM monitor.

The format of the command is:

DEFXM

This command sets the XM flag and disables the checking of location 52, bit 12. DRS-compatible programs now run by default under XXDP's extended monitor, until you issue the CLEAR command (see Section 2.3.6).

2.3.11 DELETE Command

You use the DELETE command to delete specified files.

The format of the command is:

DElete [/NOnames] dev:filnam.ext

where:

- **/NOnames** is an optional switch that prevents XXDP from printing the name of each file as it is deleted. (If you omit this switch, XXDP prints the name of each file after it is deleted.)
- **dev:** is the optional name of the device on which the file to delete resides. If you omit this parameter, the monitor searches the system device for the file to delete.

- **filnam.ext** are the name and extension of the file you want to delete.

The monitor prints the name of each file it deletes.

CAUTION: *DIGITAL does not recommend deleting files from tapes.*

The following is an example of using the DELETE command:

```
.DELETE DY0:ABC.BIN
```

This command deletes the file ABC.BIN from the device DY0:.

2.3.12 DIRECTORY Command

You use the DIRECTORY command to obtain a list of all the files on the system media.

The format of the command is:

Directory [/Printer] [/Fast]

where:

- **/Printer** is an optional switch that causes the directory to be printed on a line printer rather than the console terminal.
- **/Fast** is an optional switch that causes the directory to be printed in a short form, which consists of only the entry number and file name.

XXDP prints/displays either a long or a short directory.

- The long directory lists five items of information:
    1. The entry number
    2. The file name and extension
    3. The date the file was created, followed by the letter "C," if the file is contiguous (that is, its blocks follow one another on the media). Most files are linked; that is, their blocks are not necessarily in order on the media.
    4. The length of the file in 256 (decimal) word blocks
    5. The number of the first block of the file
- The short directory lists only file names.

The directory utility (DIR.SYS) and the read/write device driver for the system media type must reside on the system media in order for the directory command to work. If one of these files is not on the media, the monitor prints/displays an error message.

Figure 2–3 illustrates an XM long directory. Figure 2–4 illustrates an XM short directory.

Figure 2–3: XM Directory Long Form

```
ENTRY#  FILNAM.EXT        DATE         LENGTH     START
     1  XXDPxM.SYS     02-JUN-79         12       000100
     2  DY    .SYS     02-JUN-79          5       000120
     3  DIR   .SYS     02-AUG-79          6       000066
```

Figure 2–4: XM Directory Short Form

```
1  XXDPSM.SYS
2  DY    .SYS
3  DIR   .SYS
```

2.3.13 ENABLE Command

NOTE: *The ENABLE command is valid only for multidrive devices and affects drives, not controllers.*

You use the ENABLE command to designate a different drive as the system (default) device without rebooting the system. For example, if you booted the system from drive 0 of an RX02 and now want the monitor to use drive 1 as the system (default) device, use this command to make the change without rebooting the monitor.

The format of the command is:

E drive_number

where:

- **drive_number** is the number of the drive you want to enable.

A valid XXDP V2 media must reside on the drive you specify or the system does not complete the command and prints/displays an invalid device error. (See Appendix G.)

The following is an example of using the ENABLE command:

```
.E 1
```

This command enables drive 1 as the system device.

2.3.14 HELP Command

You use the HELP command to obtain a summary of XXDP commands.

The format of the command is:

Help

XXDP prints/displays a command summary and topic prompt. The file named HELP.TXT must reside on the system media.

The following is an example of using the HELP command:

```
.H
```

2.3.15 INITIALIZE Command

CAUTION: *The INITIALIZE command destroys all data residing on the media, including customer data.*

You use the INITIALIZE command to clear and initialize a device directory.

The format of the command is:

Initialize device

where:

- **device** is the name of the device you want to initialize. There is no default for the device. The device must be on-line and write-enabled.

XXDP initializes a media by clearing the bit map (random access devices) or writing an end-of-tape mark (sequential access devices) and by placing an empty directory on the media.

The monitor prints a warning when you invoke this command. The warning states which device is involved and asks you to confirm the command.

The following is an example of using the INITIALIZE command:

```
.INI DY1:
USER DATA ON DY1 WILL BE DESTROYED! .... PROCEED?  (Y/N/CR=N)
```

The only answer that confirms your wish to proceed is "Y."

If you specify the system device in the command, the monitor prints an additional warning, as in the following example.

```
INITIALIZE SYSTEM DEVICE .... PROCEED?  (Y/N/CR=N)
```

The only answer that confirms your wish to proceed is "Y."

2.3.16 LOAD Command

You use the LOAD command to load a file into memory. The LOAD command loads a program into memory, but XXDP does not start the program.

The format of the command is:

Load filnam[.ext]

where:

- **filnam** is the name of the file you want to load.
- **ext** is the optional extension of the file that you want to load. If you omit this parameter, XXDP searches for a file of the same name with a .BIN or .BIC extension. If the media contains a file with the specified name and both extensions, XXDP uses the first file that it finds.

After you issue this command, XXDP prints/displays the full file name of the loaded program. You can think of the LOAD command as the first half of a RUN command (see Section 2.3.19).

The following are examples of using the LOAD command:

```
❶ .L DIAG
❷ .L ZDJCA2.NEW
```

❶ Load DIAG.BI?

❷ Load ZDJCA2.NEW

2.3.17 PRINT Command

You use the PRINT command to print the contents of a file on the system line printer.

The format of the command is:

Print [/NORewind] [dev:]filnam.ext

where:

- **/NORewind** (tape devices only) is an optional switch that prevents XXDP from rewinding the tape between files when printing multiple files. If you omit this switch, XXDP rewinds the tape after it prints a file.
- **dev:** is the optional name of the device on which the file resides. If you omit this parameter, XXDP searches for a file of that name on the system device.

- **filnam.ext** are the name and extension of the file you want to print. The file is typically a text file.

The following are examples of using the PRINT command:

```
❶ .PRINT DU0:SYSTEM.CCC
❷ .P/N MM1:*.TXT
```

❶ The monitor prints the file SYSTEM.CCC residing on device DU0: on the system line printer.

❷ The monitor prints all files with the .TXT extension that reside on tape drive MM1: on the system line printer. The monitor does not rewind the tape after it prints each file.

2.3.18 RENAME Command

You use the RENAME command to change the name and/or extension of an existing file.

The format of the command is:

REname dev:ofilnam.ext dev:nfilnam.ext

where:

- **dev** is the name of the device on which both the old and new files reside. The device must be the same. It is assumed to be on-line and write-enabled. You cannot rename files residing on a tape device.
- **ofilnam.ext** are the name and extension of the file you want to rename (the *old* file).
- **nfilnam.ext** are the new name and extension of the file.

The monitor changes the name of the file as recorded in the directory but does not change data contained in the file.

The following is an example of using the RENAME command:

```
.RENAME DY1:DIAG.OLD DY1:DIAG.BIN
```

This command renames the file DIAG.OLD on DY1 to DIAG.BIN.

2.3.19 RUN Command

You use the RUN command to load and start a program that is stored on the load, or system, media. The program must be an executable file.

NOTE: *The RUN command combines the LOAD (see Section 2.3.16) and START (see Section 2.3.21) commands.*

The format of the command is:

Run filnam[.ext] [addr]

where:

- **filnam** is the name of the file you want to run.
- **.ext** is the optional file extension. If you omit this parameter, XXDP searches for a file of the same name with a .BIN or .BIC extension. If the media contains a file with the specified name and both extensions, XXDP uses the first file that it finds.

  You can give wildcard characters in the file specification. The monitor loads and runs the first file found that fits the wildcard description.
- **addr** is the program's optional transfer address (octal). XXDP starts the file at 200 octal if you do not indicate a transfer address. You can specify a starting address if you want the file to start at a different point.

After XXDP finds and loads the program, it prints/displays its full name so that you can verify the load before XXDP starts the program. You can use this report to check that the right program will be run, especially if you have specified a wildcard.

The following are examples of using the RUN command:

❶ `.R UPDAT`

❷ `.R SAMPLE.XXX`

❸ `.R RXDIAG 204`

❶ Load and start UPDAT.BI?

❷ Load and start SAMPLE.XXX

❸ Load and start RXDIAG.BI? at location 204

2.3.20 SET Command

You use the SET command to change device handler characteristics and certain system configuration parameters.

The format of the command is:

SEt [physical-device-name:] condition

where:

- **[physical-device-name:]** is the name of the device handler whose characteristics you want to change.
- **condition** is the system parameter that you want to change. The only system parameter that is changed by the SET command is TT:. You can change the following characteristics:
  - **[NO]QUIET**

    QUIET prevents the system from echoing lines from the chain file or from diagnostics that may be running from a chain file (providing the diagnostic has UFD support).

    NOQUIET echoes lines from batch files or running diagnostics invoked by a batch file. The default is **NOQUIET**.
  - **[NO]SCOPE**

    SCOPE deletes RUBOUT characters. This is the normal mode when you boot the extended monitor.

    NOSCOPE echoes RUBOUT characters by enclosing the deleted characters in backslashes. This is the normal mode when you boot the XXDP small monitor.

The following are examples of using the SET command:

❶ `.SET TT:SCOPE`

❷ `.SET TT:NOSCOPE`

❸ `.SET TT:QUIET`

❹ `.SET TT:NOQUIET`

❶ The monitor now echoes RUBOUT characters as backspace-space-backspace.

❷ The monitor now echoes RUBOUT characters by enclosing the deleted characters in backslashes.

❸ The monitor now does not echo lines from the chain file or from diagnostics that may be running from a chain file.

❹ The monitor now echoes lines from chain files or from diagnostics that are running from a chain file.

2.3.21 START Command

CAUTION: *Do not issue other commands between the LOAD and START commands. You may overwrite the program you have loaded.*

You use the START command to start a file that you have already loaded into core memory using the LOAD command.

The format of the command is:

Start [addr]

where:

- **addr** is the program's optional transfer address (octal). XXDP starts the file at 200 octal if you indicate no transfer address. You can specify a starting address, if you want the file to start at a different point.

The LOAD and START command sequence lets you load a program and halt the processor so that you can modify memory contents. You then restart the program.

The following are examples of using the START command:

```
❶ .L RXDIAG
❷ .S

❸ .L RXDIAG
❹ .S 204
```

❶ Load RXDIAG.BI?

❷ Start at transfer address

❸ Load RXDIAG.BI?

❹ Start at 204

2.3.22 TEST Command

You use the TEST command to enter user friendly mode.

The format of the command is:

.TEST

XXDP displays its introductory and main test menus. You can now run the XXDP extended monitor through menus that prompt you for input. Consult your PDP-11's systems troubleshooting guide for detailed information on running XXDP through user friendly menus.

2.3.23 TYPE Command

You use the TYPE command to display the contents of a file on the console terminal screen.

The format of the command is:

Type [/NORewind] filnam.ext

where:

- **/NORewind** (tape devices only) is an optional switch that prevents XXDP from rewinding the tape between files when typing multiple files. If you omit this switch, XXDP rewinds the tape after it types the file.
- **filnam.ext** are the name and extension of the file you want typed. You can include wildcards in the file name. If you leave out the file name, the monitor prompts you for this information.

2.3.24 VERSION Command

You use the VERSION command to find the version number of the extended monitor.

The format of the command is:

VErsion

Chapter 3
Diagnostic Runtime Services

The diagnostic runtime services (DRS) are an extension of the XXDP runtime monitor. They control compatible diagnostic programs. When you run a DRS-compatible diagnostic, XXDP automatically loads and starts DRS. DRS also provides diagnostics with nontest-related services, such as console terminal support.

The material in this chapter is presented in five parts:

1. DRS description (Section 3.1)
2. Commands (Section 3.2)
3. Switches or command qualifiers (Section 3.3)
4. Operational flags (Section 3.4)
5. Table building process (Section 3.5)

3.1 Description

XXDP uses two DRS systems:

- DRSSM–a small DRS system for the XXDP small monitor or the small monitor emulator
- DRSXM–an extended DRS system for the XXDP extended monitor

DRSXM allows larger diagnostics to run than DRSSM does and it provides additional functions.

All DRS-compatible programs respond to the same general set of DRS commands and report errors and gather hardware and operational data in the same way.[1]

[1] If you do not know which diagnostic programs that are residing on a media are DRS-compatible, use the SETUP utility to list the diagnostics. Chapter 6 describes SETUP.

3.1.1 DRS Start-up

You start DRS by issuing a RUN command to run a diagnostic program. XXDP responds by loading and starting the program. The diagnostic transfers control back to the XXDP monitor, which loads the appropriate DRS from the system media.[1] DRSXM uses memory management to size the memory before it enters command mode. If the hardware involved has problems, DRS does not start properly, and you must run memory management diagnostics.

3.1.2 DRS Concepts

Keep in mind these DRS concepts:

- **Console Commands**. You communicate with DRS by entering commands at the console terminal keyboard. DRS communicates with you by displaying messages on the console terminal screen or printing them on the console printer.
- **Switches (Command Modifiers) and Flags**. You can add a switch(es) to a command to alter its effects. For example, most commands affect all units (devices) that a diagnostic can test. You can specify a switch to limit the effect of commands to only certain units.

  You specify flags to set up operational parameters. For example, the LOE (loop on error) flag causes a diagnostic to continually reexecute the code that encountered an error.
- **Units**. The diagnostic acts on specified hardware. Each individual hardware entity is referred to as a unit under test (UUT) or simply a unit. DRS handles up to 64 units. You refer to a unit by a number. Units are numbered according to the order in which they were specified in the hardware tables that are part of each diagnostic. The first unit is 0.
- **Hardware Parameter Tables**. DRS-compatible diagnostics do not determine hardware information by performing bus-related tests (autosizing). Diagnostics are table-driven, that is, a diagnostic's hardware parameter tables store this information for use by XXDP. There is one table for each unit to be tested, and all information about a hardware unit is contained in this table.

  You give the diagnostic the necessary information about the hardware before running the program. The information required depends on the diagnostic. When you construct the table, the diagnostic prompts you for the information it needs for each unit, starting with unit 0.

[1] The DRS file is DRSSM.SYS (small DRS) or DRSXM.SYS (large DRS).

Section 3.5 discusses building tables.

- **Software Parameter Table**. You can specify operational parameters that affect the way a diagnostic functions. You supply these specifications to a data table called a software parameter table. Section 3.5 discusses building tables.
- **Pass**. The unit of diagnostic operation is a pass, that is, the execution of all specified tests on all active UUTs.
- **Test**. DRS diagnostics are divided into independent structures called tests. You can run all tests in a diagnostic or select a desired subset.

3.1.3 Error Messages

When a diagnostic detects an error in a UUT, it calls upon DRS to report the error to the operator. Error messages are divided into levels. You can determine before you run a diagnostic what portion(s) of the error report DRS will display/print. Section 3.4 describes the flags that control error reporting.

There are three levels of error messages:

1. **Header**–supplies general information about the error, as shown in the example below:

   ```
   ZNAME HRD ERR 00002 ON UNIT 5 TST 012 SUB 000 PC:013134
   ```

   The information in the header is as follows:

   - Diagnostic name–ZNAME
   - Error type–HRD
   - Error identification number–00002
   - Unit number–5
   - Test number–12
   - Subtest number–0
   - Location of error call to DRS–013134

2. **Basic**–gives a short, simple description of the error, as in the example below:

   ```
   REGISTER FAILED TO CLEAR AFTER BUS RESET
   ```

3. **Extended**–reproduces header- and basic-level messages and typically gives supporting information such as register contents at the time of the error. The following is an example of an extended-level message:

```
ZNAME HRD ERR 00002 ON UNIT 5 TST 012 SUB 000 PC:013134
REGISTER FAILED TO CLEAR AFTER BUS RESET
CSR:  000000 SCSR:  010000 ERRREG:  000000
```

   The first line is the header message, the second is the basic message, and the third is the extended message.

3.2 Commands

You enter DRS commands in response to the DRS prompt: **DR>**.

DRS prints/displays the prompt after it enters command mode, that is, after it has:

- Been successfully loaded (see Section 3.1.1)
- Completed running specified diagnostic operations
- Detected an error during a halt-on-error (HOE) sequence
- Been interrupted because you entered a Ctrl/C

You need enter only the first three characters of a command. For example, you can enter the START command by typing STA, STAR, or START.

You modify command effects by specifying the switches described in Section 3.3.

Table 3–1 lists DRS commands by their functions.

Table 3–1: DRS Commands

Command	Function
Execution	
CONTINUE	Continue diagnostic at test that was interrupted by a Ctrl/C
PROCEED	Continue from an error halt
RESTART	Start diagnostic and do not initialize
START	Start diagnostic and initialize

Table 3–1 (Cont.): DRS Commands

Command	Function
Data Collection	
REDIRECT	Redirect error messages and statistics to another unit and/or the line printer[1]
Units Under Test	
ADD	Activate a unit for testing
DROP	Deactivate a unit
DISPLAY	Print a list of device information
Flags	
FLAGS	Print status of all flags
ZFLAGS	Reset (clear) all flags
Exiting	
EXIT	Return to XXDP monitor

[1]DRSXM only.

Section 3.2.1 through Section 3.2.11 discuss the commands in detail.

3.2.1 ADD Command

You use the ADD command to activate a selected unit or all units for testing.

The format of the command is:

ADD [/UNIts:unit-number]

where:

- **/UNIts:unit-number** is an optional switch specifying the number of the unit(s) you want to activate. Section 3.1.2 describes the **/UNIts** switch in detail. If you omit this switch, DRS returns all deactivated units to active testing.

All units are initially active by default and must already have been deactivated by the user (see Section 3.2.4) or the diagnostic.

3.2.2 CONTINUE Command

You use the CONTINUE command to resume diagnostic operation after you have suspended it by entering a Ctrl/C or after a diagnostic has responded to a halt-on-error flag by suspending execution. (See Section 3.4.)

The format of the command is:

CONtinue [switch-list]

where:

- **switch-list** is any combination of valid switches. Table 3–3 lists valid switches.

DRS restarts the diagnostic at the beginning of the test that was interrupted. However, before DRS resumes testing, it gives you the opportunity to select new operational parameters by changing the software table. You cannot change the hardware tables.

Testing runs for the number of passes remaining in the pass count specified in the last START or RESTART command, unless you have specified the /PASS switch to change the number. All flags remain set or cleared as previously specified.

If you want to run the diagnostic again by reinitializing it and testing from the first test of the first unit, use the START command. (See Section 3.2.10.) If you want to run the diagnostic again by partially reinitializing it and testing from the first test of the first unit, use the RESTART command. (See Section 3.2.9.)

The following is an example of using the CONTINUE command:

```
❶ ^C
❷ DR>CON
❸ CHANGE SW (L) ? N
```

❶ The operator types Ctrl/C to halt the diagnostic.

❷ The operator issues the CONTINUE command.

❸ DRS restarts the diagnostic and asks if the operator wants to change the software table.

3.2.3 DISPLAY Command

You use the DISPLAY command to examine the contents of the hardware tables.

The format of the command is:

DISplay[/UNITS:unit_list]

where:

- **/UNIts:unit-number** is an optional switch specifying the number of the unit(s) you want to test. Section 3.3.5 discusses this switch.

DRS prints/displays all table data concerning the specified units on the console terminal. Units that have been dropped are so designated.

DRS prints/displays all units described in the hardware tables, unless you have specified the /UNIts switch to change the number.

3.2.4 DROP Command

You use the DROP command to deactivate a unit from testing.
The format of the command is:

DROp [/UNIts:unit-number]

where:

- **/UNIts:unit-number** is an optional switch specifying the number of the unit(s) you want to drop. All units are initially active.

3.2.5 EXIT Command

You use the EXIT command to exit from DRS and return to the XXDP monitor.

The format of the command is:

EXIT

3.2.6 FLAGS Command

You use the FLAGS command to find the current status of the DRS flags.

The format of the command is:

FLAgs

DRS responds by printing/displaying the status of all flags on the console terminal. Section 3.3.2 explains how to set flags.

The following are examples of using the FLAGS command:

❶
```
DR>FLA
FLAGS SET
NONE
```

❷
```
DR>FLA
FLAGS SET
IER
LOE
```

❶ DRS responds to the command by indicating that no flags have been set.

❷ DRS responds to the command by indicating that two flags have been set, IER (inhibit error reports) and LOE (loop on error). Section 3.4.7 and Section 3.4.10 describe these flags.

3.2.7 PROCEED Command

You use the PROCEED command only with DRS's halt-on-error feature. When you have set the HOE flag and the diagnostic encounters an error, DRS returns to command mode.

The format of the command is:

PROceed [/FLAgs:flag-list]

where:

- **/FLAgs:flag-list** is an optional switch you use to set DRS operational flags. Section 3.3.2 describes the /FLAGS switch.

The flags that you have set or cleared remain set or cleared, unless you specify this switch to change operational characteristics.

DRS does not access the UUT, reinitialize the diagnostic, or change the corresponding vector space.

Enter the PROCEED command to restart testing at the point at which the diagnostic reported the error. The command lets you examine the state of the unit and then continue testing without disturbing diagnostic operation.

3.2.8 REDIRECT Command

NOTE: *Only DRSXM (the extended monitor) supports the REDIRECT command.*

You use the REDIRECT command to redirect information from the console terminal to another device or to cancel redirection.

The format of the command is:

REDirect [/DEV] [/LPT] file-spec

where:

- **/DEV** is an optional switch naming the device on which the data is to be stored. If you omit this switch, the data is stored on the system media. The device's driver must reside on the system media.
- **/LPT** is an optional switch that directs DRS to print the information on the system line printer in addition to collecting it on the device specified.

You can use either or both switches in any order.

You can cancel the REDIRECT command at any time by entering it without arguments.

The data is collected in a file named COLECT.DAT. If this file is not present, DRS creates it and writes the data to it. If the file is present, DRS appends the data to it. DRS opens the file at the beginning of each diagnostic pass and closes it at the end. It also closes the file when it returns to command level, as when it encounters a trap or you enter Ctrl/C. It updates the date of the file to the current date.

The information DRS normally prints/displays, includes:

- Error calls, error block calls, and all diagnostic-code PRINT calls that are within the scope of that error call (for example, the error subroutine, extended error message, and so on).
- PRINT calls
- The end-of-pass printout

DRS does not save information contained in PRINT calls that are not within the scope of the error. DRS outputs it on the console terminal.

The following is an example of using the REDIRECT command:

```
DR> RED/DEV:DY0:/LPT
```

The user redirects diagnostic runtime information to disk DYO:, where it is stored in COLECT.DAT at the same time as it is printed on the lineprinter.

3.2.9 RESTART Command

You use the RESTART command to start a diagnostic after it has halted and to begin testing from the first test of a test suite.

The format of the command is:

REStart [switch-list]

where:

- **switch-list** is any combination of valid switches. Table 3–3 lists valid switches.

Unless you have specified switches to change these characteristics:

- DRS runs all tests on all units.
- DRS clears all flags.
- Testing continues until interrupted by a Ctrl/C or system error.
- DRS prints an end-of-pass message after completing each pass.

After you issue the RESTART command, DRS reinitializes the diagnostic, but does not change the diagnostic's vector space. A diagnostic may initialize itself differently after you issue the RESTART command than after you issue a START command. Refer to the diagnostic's documentation for details.

Before testing is resumed, DRS gives you the opportunity to select new operational parameters by changing the software table.

The following is an example of using the RESTART command:

```
DR> RES
CHANGE SW (L) ?  N
```

3.2.10 START Command

You use the START command to start a diagnostic from its initial state. It is the first command you issue to DRS.

The format of the command is:

STArt [switch-list]

where:

- **switch-list** is any combination of valid switches. Table 3–3 lists valid switches.

DRS responds by executing all the diagnostic's initialization code. Refer to the diagnostic's documentation for the exact nature of the initialization process carried out.

Unless you have specified switches to change these characteristics:

- DRS runs all tests on all units.
- DRS clears all flags.
- Testing continues until interrupted by a Ctrl/C or system error.
- DRS prints an end-of-pass message after each pass.

After you issue the START command, XXDP asks if you want to change the hardware information. Answer Y for yes

- If there are no existing hardware tables. Hardware tables already exist if they were entered by:
  - a previous START command sequence
  - the SETUP utility
  - a programmer who hardcoded tables into the diagnostic image
- If you want to override existing tables

XXDP then prompts for the number of units to be tested. Enter the decimal number of units.

XXDP then asks for hardware-specific information for each unit, according to the design of the diagnostic. Section 3.5 gives examples of answering hardware questions. (If you have difficulty answering, refer to the diagnostic's documentation or direct your questions to Low End Diagnostic Engineering.)

DRS then asks if you wish to change the operational data (by altering the software table), as in the following example:

```
CHANGE SW (L)  ?  N
```

Answer Y only if you want to modify default operational characteristics. Refer to the diagnostic's documentation for answers to specific questions.

The following are examples of using the START command:

❶
```
DR> STA
 CHANGE HW (L) ?  Y
 UNITS (D) ?  N
```

❷
```
DR>STA
 CHANGE HW (L) ?  N
 CHANGE SW (L) ?  N
 NO UNITS
 DR>
```

❶ XXDP asks for the hardware data for "n" units, where "n" is a decimal number between 1 and 64.

❷ Hardware tables are not present and DRS displays/prints an error message.

3.2.11 ZFLAGS Command

You use the ZFLAGS command to clear all DRS flags.

The format of the command is:

ZFLags

3.3 Switches

You add switches to a command to modify its effect. For example, the START and RESTART command act on all units listed in a diagnostic's hardware table. However, you can add the /UNITS switch to the command to limit testing to only the units you want to test.

Table 3–2 lists DRS switches and their functions. Section 3.3.1 through Section 3.3.6 discuss DRS switches in detail.

Table 3–2: DRS Switches and Their Functions

DRS Switch	Function
/EOP:ddddd	Report end-of-pass after each ddddd pass (ddddd = 1 to 65536)
/FLAGS:flag-list	Set specified flags
/TESTS:test-list	Execute only the tests specified
/PASS:ddddd	Execute ddddd passes (ddddd = 1 to 65536)
/UNITS:unit-list	Run command only on specified units

You cannot use all switches with all commands. For example, when you issue the PROCEED command to restart a diagnostic after it has halted-on-error, you can reset the flags you previously specified by adding the /FLAGS switch to the command. However, you cannot add the switches that change the tests that run, the number of passes that are executed, or the number of units that are tested.

Table 3–3 lists which switches you can use with each command.

Table 3–3: DRS Switches and Commands

Command	/EOP	/Flags	/Pass	/Tests	/Units
ADD					X
CONTINUE	X	X	X		
DISPLAY					X
DROP					X
EXIT					
FLAGS					
PROCEED		X			
REDIRECT					
RESTART	X	X	X	X	X
START	X	X	X	X	X
ZFLAGS					

3.3.1 /EOP Switch

You use the /EOP switch with the CONTINUE, RESTART, and START commands to specify when DRS display/prints end-of-pass messages. These messages indicate the number of passes completed and the number of errors found.

The format of the switch is:

/EOP:number-passes

where:

- **number-passes** is a decimal number between 1 and 65536. DRS prints an end-of-pass message each time the number of passes specified is completed. If you do not specify this switch, DRS prints a message after every pass.

The following is an example of using the /EOP switch to print the message after every 90 passes:

```
DR>RES/EOP:90
```

3.3.2 /FLAGS Switch

You use the /FLAGS switch with the CONTINUE, PROCEED, RESTART, and START command to set DRS operational flags.

The format of the switch is:

/FLAgs:flag-list

where:

- **flag-list** is a list of DRS flags separated by colons. Section 3.4 describes these flags in detail.

DRS clears all flags, unless you add the /FLAGS switch to the above commands.

The following are examples of using the /FLAGS switch:

```
❶  DR>STA/FLAGS:LOE

❷  DR>RES/FLA:LOE:IER:BOE
```

❶ This command starts a diagnostic and specifies the LOE flag to direct it to loop on error.

❷ This command restarts a diagnostic and specifies the LOE flag to direct it to loop on error, the IER flag to inhibit error reporting, and the bell-on-error (BOE) flag so that DRS rings the terminal bell when it finds an error.

3.3.3 /PASS Switch

You use the /PASS switch with the CONTINUE, START, and RESTART commands to specify the number of passes that a diagnostic will run. One pass consists of all specified tests on all active units.

The format of the switch is:

/PASs:number-passes

where:

- **number-passes** is a decimal number between 1 and 65536 specifying the number of passes a diagnostic runs.

DRS runs the diagnostics indefinitely, unless you add the /PASS switch to the commands.

The following are examples of using the /PASS switch:

```
DR>STA/PASS:100
DR>RES/PAS:1
```

3.3.4 /TESTS Switch

You use the /TESTS switch with the RESTART and START commands to specify which tests contained in a diagnostic program DRS will run.

The format of the switch is:

/TESts:test-list

where:

- **test-list** is a list of test numbers separated by colons. Test numbers indicate the tests contained in one diagnostic. If the test numbers are sequential, you can specify them by entering the first and last test numbers separated by a hyphen. You can enter test numbers in any order, but DRS executes them in numeric order.

DRS runs all tests in the diagnostic, unless you add the /TESTS switch to the commands.

The following are examples of using the /TESTS switch:

```
❶ DR>START/TESTS:5
❷ DR>START/TES:1:2
❸ DR>RES/TES:1:5-9:15
```

❶ The user started test 5 only.

❷ The user started tests 1 and 2.

❸ The user restarted the diagnostic, running only tests 1, 5, 6, 7, 8, 9, and 15.

3.3.5 /UNITS Switch

You use the /UNITS switch with the ADD, DISPLAY, DROP, RESTART, and START commands. The switch specifies which available units DRS tests.

The format of the switch is:

/UNIts:units-list

where:

- **units-list** is a list of unit numbers separated by commas. If the units are sequential, you can specify them by entering the first and last test numbers separated by a hyphen. Unit numbers are decimal numbers from 1 to 64. A unit is assigned a number based upon order of entry into the tables. The first unit is unit 1.

DRS tests all units in the scope of the command, unless you add the /UNITS switch to the commands.

The following are examples of using the /UNITS switch:

```
❶ DR>DRO/UNITS:1
❷ DR>ADD/UNI:2,3
❸ DR>RES/UNI:5-9
```

❶ This command and switch drop unit 1 from testing.

❷ This command and switch add units 2 and 3 to the units to be tested.

❸ This command and switch restart the diagnostic and test units 5, 6, 7, 8, and 9.

3.3.6 Combining Switches

You can specify as many valid switches with a command as you wish and in any order you wish.

The following is an example of combining switches:

```
DR>START/UNITS:1-4/TESTS:1:5:15/PASS:100/EOP:10
```

This command starts a diagnostic and:

- Tests units 1 through 4 only
- Runs tests 1, 5, and 15
- Executes 100 passes
- Reports the end-of-pass data after every 10 passes

3.4 Flags

When you issue the CONTINUE, PROCEED, RESTART, and START commands, you set flags to establish certain operational parameters. Unless you do so, DRS automatically clears all flags before executing these commands.[1]

You set flags using the /FLAGS switch (see Section 3.4). For example, if you want a diagnostic to loop at the point at which it encounters an error, you use the /FLAGS switch to set the LOE flag before running the diagnostic.

[1] The ZFLAGS command also clears all flags.

Table 3–4 lists DRS flags and their effects on commands. Section 3.4.1 through Section 3.4.14 discuss DRS flags in detail.

Table 3–4: Flags and Their Effects

Flag	Effect
ADR	Execute autodrop code (on diagnostics which have evaluation support)
BOE	Ring bell on error
EVL	Execute evaluation (on diagnostics which have evaluation support)
HOE	Halt on error (control is returned to runtime services command mode)
IBE	Inhibit all error reports except first level (first level contains error type, number, PC, test, and unit)
IDU	Inhibit program dropping of units
IER	Inhibit all error reports
ISR	Inhibit statistical reports (does not apply to diagnostics that do not support statistical reporting)
IXE	Inhibit extended error reports (those called by PRINTX macros)
LOE	Loop on error
LOT	Loop on test
PNT	Print test number as test executes
PRI	Direct messages to line printer
UAM	Unattended mode (no manual intervention)

3.4.1 ADR Flag (AutoDRop)

You set the ADR flag to execute a diagnostic's autodrop code. This code tests devices for device-ready or device-available status. If the unit being tested is not ready or available, DRS drops it from testing. Not all diagnostics have autodrop code. Refer to the diagnostic's documentation to find out if a diagnostic uses autodrop code.

3.4.2 BOE Flag (Bell On Error)

You set the BOE flag to ring the terminal bell when a diagnostic encounters an error. You normally set this flag when you have set flags to inhibit message reporting. See Section 3.4.7.

3.4.3 EVL Flag (EVaLuate)

You set the EVL flag to execute diagnostic evaluation code. Not all diagnostics use this type of code. Refer to the diagnostic's documentation for details.

3.4.4 HOE Flag (Halt On Error)

You set the HOE flag to execute a halt-on-error sequence when the diagnostic detects an error. The diagnostic is suspended but the processor does not halt. The exact process is:

1. The error is reported to DRS, and it prints/displays the corresponding message(s), unless printing has been inhibited.
2. It suspends the diagnostic at the point at which the error was reported.
3. The unit is left in the state that it was in at the time of the call.
4. DRS returns to command mode, issues its prompt, and waits for further input.

You can now issue a PROCEED command to resume diagnostic execution at the point at which it was suspended. You can also issue other commands as desired.

3.4.5 IBE Flag (Inhibit Basic Errors)

You set the IBE flag to inhibit basic- and extended-level reporting. DRS prints/displays only the first (header) level of messages. Section 3.1.3 discusses message levels and contents.

3.4.6 IDU Flag (Inhibit Dropping of Units)

You set the IDU flag to inhibit a diagnostic from dropping units. Diagnostics can drop a unit from testing if more than a given number of errors is reported or if a serious error is detected. You typically set this flag to keep the unit selected in order to trace the error.

3.4.7 IER Flag (Inhibit Error Reports)

You set the IER flag to inhibit all error reporting. No messages are printed/displayed except system error reports (such as ILL INT [illegal interrupt]) and end-of-pass reports. You typically set this flag when you set the loop-on-error flag. Setting the IER flag speeds up the test process and saves paper if you are using a hard-copy terminal.

3.4.8 ISR Flag (Inhibit Statistical Reports)

You set the ISR flag to inhibit the diagnostic from printing statistics. Not all diagnostics report statistics. Consult your diagnostic's documentation to see if it does.

3.4.9 IXE Flag (Inhibit eXtended Errors)

You set the IXE flag to inhibit extended-level error reporting. The error reports produced by the diagnostic code's PRINTX call are inhibited; however, DRS prints/displays header- and basic-level messages. Section 3.1.3 discusses message levels and contents.

3.4.10 LOE Flag (Loop On Error)

You set the LOE flag to enable error looping. Error looping means that a diagnostic continually reexecutes the code that detected the error. Looping remains in effect even if the symptoms that prompted the error report disappear. This allows for looping on intermittent errors. You stop the looping by entering a Ctrl/C, which returns DRS to command mode.

3.4.11 LOT Flag (Loop On Test)

You set the LOT flag to continually execute the test(s) you specify with the TEST switch. When you set this flag, the diagnostic does not execute its initialization and end-of-pass code.

3.4.12 PNT Flag (Print Number of Test)

You set the PNT flag to print/display the number of the test being executed.

3.4.13 PRI Flag (PRInter)

You set the PRI flag to redirect all messages to a line printer. This does not apply to command prompts.

3.4.14 UAM Flag (UnAttended Mode)

You set the UAM flag to prevent the use of manual intervention during testing.

When a diagnostic requires manual intervention, you must be present to perform actions needed for testing, such as attaching loopback connectors. You set the flag to start the diagnostic and let it run unattended. Some testing is inhibited when you set the UAM flag. Refer to a diagnostic's documentation for a description of UAM flag effects in specific cases.

3.5 Table Building

XXDP diagnostic programs have corresponding hardware and software parameter tables. These tables give the diagnostic necessary information before it runs.

Hardware tables hold information such as register addresses, drive numbers, and interrupt priority. Software parameter tables contain operational parameters affecting how a diagnostic functions, such as data patterns to use in testing.

These tables must be constructed before you start a diagnostic. They are constructed in one of the following ways:

- You use the START command to start the diagnostic.
  - The diagnostic does not yet contain tables and it prompts you to supply hardware information. When you answer the hardware questions, you are building entries in a table that describes the devices under test.
  - The diagnostic program already contains tables. You can use them as they are or change them after you issue the START command. Sometimes the tables already contain default values that you can replace with new ones. The software table is not present in all diagnostics. The diagnostic does not prompt you in this case.
- You use the SETUP utility to prebuild tables (see Chapter 6), that is, to build them without running the diagnostic and store them with the diagnostic. XXDP writes the tables into memory at the same time it activates the diagnostic.

Whether you build the table at run time or prebuild it, DRS prompts you with table-building questions. These questions have the same format:

Question (type) [default] ?

where:

- **type** is a 1-character code for the data type of the answer you give. Type codes are:
  - O for octal
  - D for decimal
  - B for binary
  - A for ASCII

- L for logical (Y or N)

You enter the data type after the question mark, as in the following example:

```
DRIVE NUMBER (d)? 0
```

If DRS cannot understand the answer, it prints/displays a message to that effect and asks for the information again.

Although you can build a table by starting a diagnostic and answering all questions relating to each UUT, the process is tedious and time-consuming if you have a multiplexed device, such as a mass storage controller with several drives or a communication device with several lines.

You can simplify table building by taking advantage of two DRS features:

- DRS uses the information you supply to build as many table entries as it can in one pass through the questions. DRS can use one specification several times.
- Null replies (commas enclosing a null field) tell DRS to repeat the last reply.

The following are examples of building hardware tables for a hypothetical device called "XY11." The device consists of a control module with eight units (subdevices) attached to it. These units are described by the octal numbers 0 through 7. The diagnostic prompts you for information about each unit. There is one hardware parameter that can vary among units called the Q-factor. This Q-factor may be 0 or 1. The CSR address is always 160000 (octal).

1. Example 3–1 builds the diagnostic's hardware table in eight passes (sets of prompts and responses) by answering the prompt for each subdevice.
2. Example 3–2 takes advantage of DRS's multiple specification feature and builds the table in three passes—giving information about subdevices 0 and 1, subdevices 2 through 5, and subdevices 6 and 7.
3. Example 3–3 takes advantage of DRS's ability to interpret numbers separated by a hyphen as a series and to repeat a null field. The table is built in a single pass. The user specifies subdevices 0 through 7 and then lists their respective Q-factors in a single line.

Example 3–1: Simple Table Building 1

```
UNITS (D) ?  8

UNIT 1
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  0
Q-FACTOR (O) 0 ?  1

UNIT 2
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  1
Q-FACTOR (O) 1 ?  0

UNIT 3
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  2
Q-FACTOR (O) 0 ?

UNIT 4
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  3
Q-FACTOR (O) 0 ?

UNIT 5
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  4
Q-FACTOR (O) 0 ?

UNIT 6
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  5
Q-FACTOR (O) 0 ?

UNIT 7
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  6
Q-FACTOR (O) 0 ?  1

UNIT 8
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  7
Q-FACTOR (O) 1 ?
```

Example 3–2: Simple Table Building 2

```
UNITS (D) ?  8

UNIT 1
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  0,1
Q-FACTOR (O) 0 ?  1,0

UNIT 3
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  2-5
Q-FACTOR (O) 0 ?  0

UNIT 7
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  6,7
Q-FACTOR (O) 0 ?  1
```

Example 3–3: Simple Table Building 3

```
UNITS (D) ?  8

UNIT 1
CSR ADDRESS (O) ?  160000
SUB-DEVICE   (O) ?  0-7
Q-FACTOR (O) 0 ?  0,1,0,,,,1,1
```

Chapter 4
UPDAT Utility

You use UPDAT, a file manipulation utility program, to do the following:

- Build XXDP media
- Copy files from one media to another
- Delete files from a media
- Modify files
- Perform related functions

4.1 Description

UPDAT runs in the lower part of memory and occupies about 6 Kwords. It interacts with the operator and loads retrievable device drivers by means of the runtime monitor. The drivers allow it to carry out device-related functions. The drivers you intend to use must reside on the system media. With one exception (see Section 4.2.10), the system media must be on line while you use UPDAT.

4.1.1 Starting UPDAT

To start UPDAT, type:

.R UPDAT

After the monitor has successfully loaded UPDAT, it prints/displays its name and a restart address. It then prints/displays an asterisk (*) to prompt you for commands.

4.1.2 UPDAT Command Categories

UPDAT commands fall into the following categories:

- File manipulation
- File modification
- New media creation

- Print
- Miscellaneous
- Return to monitor

Table A–6 lists UPDAT commands alphabetically. Section 4.1.2.1 through Section 4.1.2.4 describe each category of commands. Section 4.2 explains how to use each command.

4.1.2.1 File Manipulation Commands

You use UPDAT file manipulation commands to maintain XXDP media. These commands allow you to:

- Transfer files from media to media
- Delete files from a media
- Rename files
- Obtain a directory of files on a media

The file manipulation commands are given in the following table:

Table 4–1: UPDAT File Manipulation Commands

Command	Function
DEL	Delete a file or files
DIR	Give directory of specified media
FILE	Transfer a file or files (with autodeletion)
PIP	Transfer a file or files (without autodeletion)
REN	Rename a file

4.1.2.2 File Modification

You use UPDAT to modify binary files. When a diagnostic program is defective, a diagnostic engineering patch order (DEPO) is issued so that you can make a temporary change to a released program.

UPDAT gives you a means of implementing temporary changes. You use it to load the defective program from an XXDP media into a memory area called the program buffer.[1] Once the program image resides in memory, you can modify it and write it back onto an XXDP media. You can also alter the program's transfer address and load image size at this time.

[1] This area lies in the physical memory space between the monitor and UPDAT. Its size equals the system size minus 8 Kwords but never exceeds 20 Kwords.

The UPDAT file modification commands are given in the following table.

Table 4–2: UPDAT File Modification Commands

Command	Function
CLR	Clear UPDAT program buffer
CORE	Print limits of buffer addresses
DUMP	Dump a program image
HICORE	Set upper memory limit for dump[1]
LOAD	Load a program
LOCORE	Set lower memory limit for dump[1]
MOD	Modify file image in memory[1]
XFR	Set transfer address

[1]In this command, locations within a program that has been loaded into the program buffer are virtual locations: their addresses are relative to the first physical location in the program buffer and not the first physical memory location.

When you use the UPDAT file modification commands, you alter the diagnostics' binary files. The extensions of these files are .BIC and .BIN.

You perform the following steps to modify a file:

1. Clear the program buffer with the CLR command
2. Load the file into the program buffer with the LOAD command
3. Change the size of the image, if necessary, with the HICORE and LOCORE commands
4. Modify the contents of the desired location(s) with the MOD command
5. Modify the transfer address, if desired, with the XFR command
6. Write the image onto the media with the DUMP command

4.1.2.3 New Media Creation

You use the UPDAT utility to create new XXDP media. Appendix D describes the build process in detail.

The new media creation commands are given in the following table.

Table 4–3: UPDAT New Media Creation Commands

Command	Function
COPY	Copy entire media
CREATE	Save a monitor on a disk or tape
ZERO	Initialize a media

4.1.2.4 Print Commands

You use UPDAT commands to print/display the text of a file on the console terminal. The printing commands are given in the following table.

Table 4–4: UPDAT Printing Commands

Command	Function
PRINT	Print a file on the line printer
TYPE	Display a file on the console terminal screen

4.1.2.5 Miscellaneous Commands

You use UPDAT commands to perform functions related to file manipulation. The miscellaneous commands are given in the following table.

Table 4–5: Miscellaneous UPDAT Commands

Command	Function
ASG	Assign a logical name to a device
DO	Execute an indirect command file
DRIVER	Load a device driver
EOT	Write logical end-of-tape mark on a tape (<EOT>)
READ	Read a file to check validity

4.1.2.6 Returning-to-Monitor Commands

You use UPDAT commands to return control to the monitor. The return commands are given in the following table.

Table 4–6: UPDAT Returning-to-Monitor Commands

Command	Function
BOOT	Bootstrap a device
EXIT	Return control to the runtime monitor

4.2 Commands

Section 4.2.1 through Section 4.2.25 describe UPDAT commands.

4.2.1 ASG Command

You use the ASG command to assign a logical unit number to a device.

The format of the command is:

ASG dev:=n

where:

- **dev:** is the name of the device to be assigned.
- **n** is the logical unit number (0–7).

You can now use the logical unit number to reference the device in UPDAT commands. You use the ASG command primarily to simplify using the DO command to run indirect command files. See Section 4.2.9.

The following is an example of using the ASG command:

```
❶  *ASG DY0:=0
❷  *PIP 0:=DY1:*.CCC
❸  *FILE 0:=MM0:FILE.CCC
```

❶ This command assigns logical unit number 0 to device DY0:.

❷ This command transfers files having a .CCC extension from DY1: to device 0.

❸ This command transfers files having a .CCC extension from MM0: to device 0.

4.2.2 BOOT Command

You use the BOOT command to start the monitor in the same manner as you use hardware bootstrap to start it. See Section 2.1.7.

The format of the command is:

BOOT dev:

where:

- **dev:** is the name of the device you want to boot. The device must have a bootable media mounted.

You can use the BOOT command to boot a device other than the original system device. The booted device is now the system device.

4.2.3 CLR Command

You use the CLR command to clear the program buffer before loading a program into it so that you can modify it.

The format of the command is:

CLR

You clear the program buffer before writing to it, so that unused locations are set to zero before a new program image is dumped to a media. If you do not clear the program buffer, unused locations may contain data from previously loaded files.

4.2.4 COPY Command

CAUTION: *Copying data destroys the existing output device data and customer data could be lost. The system always prompts you to confirm that you want to copy.*

You use the COPY command to copy the entire contents of one media to an identical media (for example, from an RP06 to an RP06).

The format of the command is:

COPY devo:=devi:

where:

- **devo:** is the output device, that is, the device to receive the copy.
- **devi:** is the input device, that is, the device whose contents you want to copy.

The devices must be of the same type. Both must be on-line, and the output device must be write-enabled.

When you issue the COPY command, UPDAT prints/displays a warning message, and copying proceeds only if you type Y for yes.

```
USER DATA ON devo WILL BE DESTROYED!
PROCEED?  (Y/N/CR=N)
```

When you issue the COPY command, UPDAT makes an image copy, that is, a block-for-block transfer. Image copies are very fast, because available memory is used as a buffer. However, if the input device has bad software blocks, the copy process may encounter a corrupted sector. If it does, XXDP displays a message to this effect and the copy process terminates without completing. You must use the PIP or FILE command (see Section 4.2.19 or Section 4.2.14) to make a file copy.

Making file copies is slower than making image copies, because only one block is transferred at a time.

4.2.5 CORE Command

You use the CORE command to find the limits of a previously loaded program's virtual memory addresses.

The format of the command is:

CORE

UPDAT prints/displays the limits of the program. You use LOCORE and HICORE to modify the program's lower and upper limits. (See Section 4.2.15 and Section 4.2.17.)

4.2.6 CREATE Command

You use the CREATE command to place a bootable monitor on a media. Appendix D discusses building XXDP media.

The format of the command is:

CREATE dev:

where:

- **dev:** is the name of the device containing the media you want to make bootable. The device must be on-line and write-enabled. The file XXDPSM.SYS and the appropriate driver must reside on the system device to build the monitor boot block image.

The command writes the appropriate secondary bootstrap to the boot block and places the monitor file on the media in a predetermined section.

You do not have to initialize the media with the ZERO command if it is already XXDP-compatible. The monitor and bootstrap code that resided there before you issued the CREATE command are lost, but all other files are preserved.

4.2.7 DEL Command

You use the DEL command to remove a file(s) from an XXDP media.

The format of the command is:

DEL dev:filnam[/N][/Q]

where:

- **dev:** is the name of the device where the file resides; the device is assumed to be on-line and write-enabled.
- **filnam** is the name of the file(s) to be deleted; the file must reside on the device specified; the extension must be specified (unless the file has no extension); wildcards are accepted.
- **/N** is an optional switch that prevents UPDAT from printing file names as they are deleted; if you omit this switch, names will be printed.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the media before searching for file(s) to delete; if you omit this switch, UPDAT rewinds the tape after the delete operation is complete.

  CAUTION: *Digital does not recommend deleting files from tape.*

The following is an example of using the DEL command:

```
*DEL DY0:XYZ001.TXT
```

This command deletes the file XYZ001.TXT on DY0:.

4.2.8 DIR Command

You use the DIR command to obtain a directory of files on a specified media. You can direct the directory to the console terminal, the line printer, or a file. You can choose a long or short directory.

The format of the command is:

DIR [[devo:][ofile][/Q]=][devi:][ifile][/Q][/F][/B][/L]

where:

- **devo:** is the optional name of the output device; the default is the console terminal unless **ofile** is specified or the equal sign (=) is used, in which case the device is the default. The device is assumed to be on-line and write-enabled.
- **ofile** is the optional name of the file to contain the directory. **devo:** must be a file-structured device. If you omit this parameter, the file name is DIR.TXT. If a file of the same name already exists, it is deleted.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the media before beginning the directory search for the file with same name as **ofile**. If you omit this switch, UPDAT rewinds the tape.
- **devi:** is the optional name of the device from which to take the directory. If you omit this parameter, a directory of the system device is taken. The device is assumed to be on-line and ready.
- **ifile** is the name of the file(s) to be listed in the directory; wildcards are legal. The default is a period (**.**).
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the media before beginning the directory search for the file with same name as **ifile**. If you omit this switch, UPDAT rewinds the tape.
- **/F** is an optional switch that causes UPDAT to give short form of directory. If you omit this switch, UPDAT gives the long form.
- **/B** is an optional switch that causes UPDAT to list the number of free blocks left on input media (random access devices only).
- **/L** is an optional switch that causes UPDAT to send the directory to a line printer (parallel printers only).

Figure 2–1 and Figure 2–2 show samples of both forms of the directory.

The following are examples of using the DIR command:

❶
```
*DIR DY0:DISK.TXT=MM0:/Q
```

❷
```
*DIR =DR1:.BIN
```

❸
```
*DIR
```

❹
```
*DIR =
```

❺
```
*DIR DY0:/F/L
```

❶ UPDAT writes a directory of files residing on MM0: to a file called DISK.TXT on DY0:. UPDAT does not rewind the tape during the operation.

❷ UPDAT writes a directory of all files with .BIN extensions residing on DR1 to a file called DIR.TXT on the system device.

❸ UPDAT prints/displays a directory of all files residing on the system device on the console terminal.

❹ UPDAT writes a directory of all files residing on the system device to a file called DIR.TXT on the system device. Please take note of the effect of the equal sign (=) on the operation of the directory command.

❺ UPDAT prints a short-form directory of all files residing on DY0: on the line printer.

4.2.9 DO Command

You use the DO command to run an indirect UPDAT command file. The format of the command is:

DO file.ext

where:

- **file.ext** are the name and extension of the indirect command file. An indirect file is a text file containing one or more UPDAT commands. The file must reside on the system device. You cannot specify a device with this command.

You use indirect command files to carry out the normal set of UPDAT commands without having to type each command individually. The only command you cannot include in an indirect file is the EXIT command. Creating and running indirect files saves you time. You can create a command file that accomplishes some common task, such as building new media.

The following rules apply to creating command files:

- UPDAT treats a command line that begins with a semicolon (;) as a comment and takes no action. UPDAT merely prints the line.
- UPDAT similarly treats a command line beginning with a dollar sign ($) as a comment. However, UPDAT stops processing the command file after the line is printed and resumes processing only after you type a Ctrl/X. This function can be used to stop activity while the operator

performs some required task such as mounting a new media or placing a device on line.

You can make a command file more global in scope than it would be if you used physical device names. You can save time by using the ASG command (see Section 4.2.1) to assign logical unit numbers to physical devices before you run the indirect command file.

The following is an example of an indirect command file:

```
;Sample Command File:  RMBLD.TXT

ZERO 1:
Y
CREATE DR1:
FILE 1:=0:*.SYS
FILE 1:=0:UPDAT.BIN
```

This indirect file builds the XXDP System on any RM02/3 from any other XXDP media. The command line containing a "Y" is required only to verify the zero process. (Section 4.2.25 describes the ZERO command.) Before you run RMBLD.TXT, you assign logical unit numbers to the device specified in the FILE commands.

The following is an example of activating UPDAT and using the DO command to run the indirect file RMBLD.TXT:

```
. R UPDAT
*ASG DR2:=1
*ASG MM0:=0
*DO RMBLD.TXT
*EXIT
```

4.2.10 DRIVER Command

You use the driver command to explicitly load a read/write device driver into memory.

The format of the command is:

DRIVER driver[/driver]

where:

- **driver** is the 2-character device name, for example, DY: for an RX02.

You can load up to two drivers. If you load a third, the driver that was loaded in memory first is deleted. If you request a driver that already resides in memory, UPDAT takes no action.

Appendix B lists supported devices.

Issue the DRIVER command to build an XXDP media using limited resources. If the system device is required for building a new media, the user can load the drivers required, remove the system media, mount the new media, and build XXDP.

The following is an example of using the DRIVER command:

```
*DRIVER DY:/DK:
```

4.2.11 DUMP Command

You use the DUMP command to write the program image located in the program buffer to a file on a media.

The format of the command is:

DUMP [dev:]ofile[/Q]

where:

- **dev:** is the optional name of the device to which the file is written. If you omit this parameter, UPDAT writes the image to the system device. The device is assumed to be on-line and write-enabled.
- **ofile** is a unique file name to be given to the binary file; wildcards are not accepted.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the media before searching for a logical end-of-tape. If you omit this switch, UPDAT rewinds the tape.

You find the image size by examining the upper and lower memory limits displayed by the HICORE and LOCORE commands. (See Section 4.2.15 and Section 4.2.17.) UPDAT puts a transfer address into the file. You can use the XFR command to examine this address and alter it if need be. (See Section 4.2.24.)

The following are examples of using the DUMP command:

```
❶ *DUMP DY0:ZRLAA1.BIN

❷ *DUMP FILE3.BIC
```

❶ The program image is written to DY0 and given the file name ZRLAA1.BIN.

❷ The image is written to the system device and given the specified name.

4.2.12 EOT Command

You use the EOT command to place a logical end-of-tape marker (<EOT>) on a tape at the current position.

CAUTION: *UPDAT does not rewind the tape, and all files after the current position are no longer accessible.*

The format of the command is:

EOT dev:

where:

- **dev:** must be a tape unit. If the system device is a tape unit, it is the default device.

4.2.13 EXIT Command

You use the EXIT command to return control to the runtime monitor.

The format of the command is:

EXIT

4.2.14 FILE Command

CAUTION: *The FILE command destroys output media files. If you do not want to destroy a file on the output media that has the same name as the transferred file, use the PIP command. (See Section 4.2.19.)*

You use the FILE command to transfer a file(s) from one media to another.

The format of the command is:

FILE [devo:][/Q]=devi:[ifile][/Q][/N]

where:

- **devo:** is the optional name of the output device—that is, the device that receives the copied file. If you omit this parameter, UPDAT searches the system device. The device is assumed to be on-line and write-enabled.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the output media before the directory search for the existing file. If you omit this switch, UPDAT rewinds the tape after every file transfer.

- **devi:** is the required name of the input device—that is, the device that holds the file to be copied. The device is assumed to be on-line and ready.

  CAUTION: *If you omit the input device name, UPDAT transfers the file from the input device to the input device, thus destroying the media.*

- **ifile** is the optional name of the input file; wildcards are permitted. If you omit this parameter, the default is a period (.). The specified file(s) must exist.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding before its directory search for the first file. If you omit this switch, UPDAT rewinds the tape after every file transfer.
- **/N** is an optional switch that prevents UPDAT from printing/displaying the name of each file as it is found. If you omit this switch, UPDAT shows each name.

The FILE command transfers the specified file(s) to the output media even if a file(s) already residing there has the same name and extension as the transferred file. The FILE command deletes from the output media any file(s) that has the same name and extension as transferred files. UPDAT does not notify the operator of this event, called autodeletion.

You cannot rename the transferred file by specifying a new file name. If you want to specify an output file name different from that of the transferred file, use the PIP command (see Section 4.2.19).

The following are examples of using the FILE command:

❶
```
*FILE DY0:=DR1:ZZZZZZ.BIN
```
❷
```
*FILE =DD0:XMONCO.LIB
```
❸
```
*FILE DU1:=DU0:
```

❶ This command transfers the file ZZZZZZ.BIN located on DR1: to DY0:. Before the transfer, it deletes a file of the same name if one already exists on DY0:.

❷ This command transfers the file XMONCO.LIB from DD0: to the system device. Before the transfer, it deletes a file of the same name, if one already exists on the system device.

❸ This command transfers all files located on DU0: to DU1:. Before the transfer, it deletes all files of the same name that already exist on DU1:. The command provides a convenient method of putting updated files onto an existing XXDP media.

4.2.15 HICORE Command

You use the HICORE command to find the address of the highest virtual memory location that is transferred during a dump operation and alter this address if needed. (The default location is printed after the LOAD command is completed.)

The format of the command is:

HICORE

UPDAT responds by printing the address of the highest virtual memory location. You can accept or change this address:

- To accept the address, press [Return].
- To change the address, type the new address (octal) and press [Return]. The address you give must be higher than that of the lowest virtual location (low core) and lower than that of the top of the program buffer.

The following are examples of using the HICORE command:

❶
```
*HICORE
100000  [Return]
```

❷
```
*HICORE
40000 45000  [Return]
```

❶ This command accepts the upper virtual location of 100000.

❷ This command changes the upper virtual location from 40000 to 45000.

4.2.16 LOAD Command

You use the LOAD command to load a binary file into memory for the purpose of modifying the program image.

The format of the command is:

LOAD [devi:]filnam.ext[/N][/Q]

where:

- **devi:** is the optional name of the device from which to load the file. If you omit this parameter, UPDAT searches the system device. The device is assumed to be on-line and ready.
- **filnam.ext** are the name and extension of the file you want to load; you can specify wildcard(s) in the file name.

- **/N** is an optional switch that inhibits printing of upper and lower memory limits; the switch inhibits printing of the file name found if you use a wildcard(s).

  CAUTION: *Specifying wildcards may lead to the loading of bad data. Unused locations in the newly loaded file may not contain zeros. The program buffer in memory is not automatically cleared between loads, and unused locations may contain data from previously loaded files.*

- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the media before searching for the file. If you omit this switch, UPDAT rewinds the media.

As UPDAT loads the file, it computes a checksum and compares it with a checksum stored with the file. If the checksums do not agree, UPDAT prints/displays a message to that effect.

After the load is successfully completed, UPDAT prints the transfer address and core limits (the lowest and highest virtual memory locations used by the program). You can use the HICORE, LOCORE, and XFR commands to change these parameters. See Section 4.2.15, Section 4.2.17, and Section 4.2.24.

You can use the LOAD command to load an XXDP monitor image into the program buffer as part of the media build process. Section 4.1.2.3 and Appendix D describe this process.

The following is an example of using the LOAD command:

```
*LOAD DY0:PROG1.BIN

XFR:  000001   CORE:  000000,020000
```

The operator loads the binary file PROG1.BIN residing on DY0: into memory. UPDAT prints/displays the transfer address and the high and low core limits.

4.2.17 LOCORE Command

You use the LOCORE command to find the address of the lowest virtual memory location that is transferred during a dump operation and alter this address if needed. (The default location is printed after the LOAD command is completed.)

The format of the command is:

LOCORE

UPDAT responds by printing the address of the lowest virtual memory location to be used during a dump operation. You can accept or change this address:

- To accept the address, press Return.
- To change the address, type the new address (octal) and press Return.

The following are examples of using the LOCORE command:

❶
```
*LOCORE
000200        Return
```

❷
```
*LOCORE
000000   20   Return
```

❶ This command accepts the low memory address.

❷ This command raises the low memory address to 20 (octal).

4.2.18 MOD Command

You use the MOD command to alter the contents of one or more virtual memory locations in a program that has been loaded by UPDAT.

The format of the command is:

MOD nnnnnn

where:

- **nnnnnn** is the octal address of the virtual memory location whose contents you want to change.

UPDAT responds by printing/displaying the address and the current contents. You can accept the contents or change them.

- To accept the contents, press Return.
- To change the contents, type the new value (octal) and press Return.

If you wish to examine or modify the next consecutive virtual memory location (<ADDR>+2), type a line feed (Line Feed) after modifying each location.

The following are examples of using the MOD command:

❶
```
*MOD 12004
012004 012736 Return
```

❷
```
*MOD 2460
002460 770   771 Return
```

❸
```
*MOD 1220
001220 120   167 [Line Feed]
001222 120   1234 [Return]
```

❶ This command examines but does not modify virtual location 12004.

❷ This command modifies virtual memory location 2460.

❸ This command modifies two consecutive virtual memory locations (1220 and 1222).

4.2.19 PIP Command

You use the PIP command to transfer a file(s) from one media to another.

The format of the command is:

PIP [devo:][ofile][/Q]=[devi:][ifile][/Q][/N]

where:

- **devo:** is the optional name of the output device—that is, the device that receives the copied file. If you omit this parameter, the system device receives the copied file. The device is assumed to be on-line and write-enabled.
- **ofile** is the file name you give the copied file.
  - If **ofile** exists on the output device, UPDAT does not perform the transfer.
  - If you omit an output file name and no file of the same name resides on the output device, UPDAT gives the output file the same name as the input file.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the output media before the directory search for the existing file. If you omit this switch, UPDAT rewinds the tape after every file transfer.
- **devi:** is the name of the input device—that is, the device that holds the file to be copied. If you omit this parameter, UPDAT searches the system device. The device is assumed to be on-line and ready.
- **ifile** is the optional name of the input file—that is, the file to be copied; wildcards are permitted. The default is a period (.). The specified file(s) must exist.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the tape before its directory search for the first file. If you omit this switch, UPDAT rewinds the tape.

- /N is an optional switch that prevents UPDAT from printing/displaying the name of each file as it is found. If you omit this switch, UPDAT shows each name.

The PIP command does not allow autodeletion—that is, the removal of a file from the output media if it has the same name and extension as the file being transferred. If you try to transfer a file whose name already exists on the output media, UPDAT does not delete the existing file or perform the transfer and it displays a message to that effect. If you use the PIP command to transfer several files, UPDAT transfers only files whose names and extensions do not match those of existing files on the output media.

NOTE: *If you want to delete files already residing on the output media that have the same name as the transferred file, use the FILE command (see Section 4.2.14).*

The following are examples of using the PIP command:

❶ `*PIP DY0:NEW.BIN=DR1:ZZZZZZ.BIN`

❷ `*PIP =DD0:XMONC0.LIB`

❸ `*PIP DU1:=DU0:`

❹ `*PIP MM0:FILE??.*=`

❶ This command copies the file ZZZZZZ.BIN from DR1: to NEW.BIN on DY0:. UPDAT does not transfer the file, however, if NEW.BIN already exists on DY0.

❷ This command transfers the file XMONC0.LIB residing on DDO: to the system device. The transferred file's name is not changed. If XMONC0.LIB already exists on the system device, UPDAT displays an error message and does not perform the transfer.

❸ This command transfers all files located on DU0: to DU1:. UPDAT does not transfer files located on DU0: if files of the same name already exist on DU1:. UPDAT displays a message to that effect and transfers the remaining files.

❹ This command transfers all files residing on the system device to MM0: and renames them so that the characters FILE replace the first four characters of the original name.

4.2.20 PRINT Command

You use the PRINT command to output file contents on a line printer.

The format of the command is:

PRINT [dev:]filnam.ext[/Q]

where:

- **dev:** is the optional name of the device where the file is located. If you omit this parameter, UPDAT searches the system device. The device must be on-line.
- **filnam.ext** are the name and extension of the file you want to print. The file must exist and contain text in ASCII format.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the media before searching for a specified file. If you omit this switch, UPDAT rewinds the tape.

The following is an example of using the PRINT command:

```
*PRINT MM1:HELP.TXT/Q
```

This command reads the file HELP.TXT residing on the tape MM1: and prints it on the line printer. UPDAT does not rewind the tape and searches for the file beginning at the current tape location.

4.2.21 READ Command

You use the READ command to check device and media integrity.

The format of the command is:

READ [dev:]filnam.ext[/N][/Q]

where:

- **dev:** is the optional name of the device from which file(s) are to be read. If you omit this parameter, files are read from the system device. The device must be on-line.
- **filnam.ext** are the name and extension of the file(s) to be read; wildcards are accepted.
- **/N** is an optional switch that prevents UPDAT from showing the name of each file as it is read.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the media before searching for specified files. If you omit this switch, UPDAT rewinds the tape.

UPDAT reads each block of the specified file into memory and calculates a checksum. It compares the checksum with the checksum stored with the file. If these checksums do not agree, UPDAT prints/displays a message to that effect.

4.2.22 RENAME Command

You use the RENAME command to change the file specification of an existing file.

The format of the command is:

REN [dev:]newnam.ext=[dev:]oldnam.ext

where:

- **dev:** is the optional name of the device where the file to be renamed is located and where it resides after the RENAME operation is completed. The device must be the same for both input and output. If you omit this parameter, UPDAT assumes that it is the system device. The device is assumed to be on-line and write-enabled.
- **newnam.ext** are the new file name and extension. A file of the same name must *not* exist on the device; wildcards are accepted.
- **oldnam.ext** are the current file name and extension. The file must exist on the device; wildcards are accepted.

UPDAT changes the specification of the file as recorded in the directory but does not change data contained in the files.

The following is an example of using the RENAME command:

```
*REN DY1:DIAG.OLD=DY1:DIAG.BIN
```

This command renames the file DIAG.BIN to DIAG.OLD on DY1:.

4.2.23 TYPE Command

You use the TYPE command to display the text of a file on the console terminal screen.

The format of the command is:

TYPE [dev:]filnam.ext[/Q]

where:

- **dev:** is the optional name of the device where the file is located. If you omit this parameter, UPDAT searches the system device. The device must be on-line.

- **filnam.ext** are the name and extension of the file that you want typed. The file must exist and contain text in ASCII format.
- **/Q** (tape devices only) is an optional switch that prevents UPDAT from rewinding the media before searching for a specified file. If you omit this switch, UPDAT rewinds the tape.

The following is an example of using the TYPE command:

```
*TYPE SYSTEM.CCC
```

UPDAT displays the file SYSTEM.CCC, which resides on the system device, on the console terminal screen.

4.2.24 XFR Command

You use the XFR command to modify the transfer address contained in the program that has been loaded.

The format of the command is:

XFR

UPDAT responds by printing the current transfer address (octal), which is listed in the file that you have created with the DUMP command. It is the address of the location to which control will be transferred when the program is started.

You can accept this address or change it.

- To accept the address, press Return.
- To change the address, type the new value (octal) and press Return.

The following are examples of using the XFR command:

```
❶ *XFR
  002000  Return
❷ *XFR
  000200   001000  Return
```

❶ This command examines the transfer address and accepts it.

❷ This command changes the transfer address from 200 (octal) to 1000 (octal):

4.2.25 ZERO Command

CAUTION: *The ZERO command destroys all data residing on the media. UPDAT does not determine what data reside on a media and may destroy customer data.*

You use the ZERO command to initialize a media by clearing it.

The format of the command is:

ZERO dev:

where:

- **dev:** is the name of the device you wish to zero. There is no default for the device. The device must be on-line and write-enabled.

UPDAT responds by issuing this warning:

```
USER DATA ON dev WILL BE DESTROYED!
PROCEED?  (Y/N/CR=N)
```

Enter Y to carry out the ZERO command.

UPDAT clears the device's bit map (random access devices) or writes an <EOT> (sequential access devices) and places an empty directory on the media.

Before you issue a ZERO command affecting the system device, you must be sure that two drivers are present in memory—one for the system device and one for the device from which files are moved to the new media in the system device. To make sure that the necessary drivers are in memory, use the DRIVER command.

UPDAT issues an additional warning if you specify the system device in the ZERO command.

```
ZERO SYSTEM DEVICE
YOU MAY NEED AN ADDITIONAL DRIVER
PROCEED?  (Y/N/CR=N)
```

Enter Y to carry out the ZERO command.

4.3 Sample Modification

The following sample file modification illustrates the use of UPDAT commands. It is based on the following patch of a hypothetical diagnostic named ZXXXB0.BIN:

Location	Old Contents	New Contents
1224	106701	240
1226	177660	240
1452	376	374

The UPDAT dialogue used to accomplish this is shown below.

Example 4–1: Sample UPDAT Modification

```
     . R UPDAT
     CHUP2B0 XXDP UPDAT UTILITY
     RESTART ADDRESS:  002432
     *LOAD ZXXXB0.BIN
     XFR:  000001 CORE:  000200, 02723
     *MOD 1224
❶    001224 106701 240 Line Feed
❶    001226 177660 240 Return
     *MOD 1224
❷     001224 000240  Line Feed
❷     001226 000240  Return
     *MOD 1452
❶     001452 000376 374 Return
     *MOD 1452
❷     001452 000374  Return
❸     *DUMP ZXXXB1.BIN
```

❶ The user changes the contents of the location.

❷ The user examines the modified location to verify the changes and accepts them.

❸ The user dumps the file with a name change that reflects the patch level.

Chapter 5
PATCH Utility

You use the PATCH utility to modify a binary formatted file (extension .BIN or .BIC) stored on an XXDP storage media. The utility gives you an alternative to the LOAD-MOD-DUMP sequence of UPDAT.

5.1 When to Use PATCH

You use PATCH if:

- You cannot use the LOAD-MOD-DUMP sequence of UPDAT, because the file you are modifying is too large for UPDAT to load into its program buffer. (The file you modify with PATCH is never completely loaded into memory.)
- You are modifying DEC/X11 runtime exercisers (RTEs) (see Chapter 8). You cannot use UPDAT to alter these programs. Instead, you use PATCH to produce a permanently modified .BIN file that is part of a DEC/X11 RTE.

5.1.1 Starting PATCH

Enter the following XXDP monitor command to start PATCH:

.R PATCH

The system loads the PATCH utility into memory. The utility identifies itself with this message:

```
PATCH   -XXDP V2 PATCH UTILITY REVISION E
```

PATCH prints/displays an asterisk (*) to prompt you for commands.

5.1.2 The PATCH Process

You patch a binary file in two phases:

1. You use PATCH commands (see Table 5–1), except the command PATCH, to build a table containing the modifications to be made to the file. This table is the input table. You fill this table with the file addresses to be modified and their new contents.

The input table may have a maximum of 50 entries.

You save the input table as a file and retrieve it for later use.

2. You use the PATCH command to combine the information in the input table with the actual binary file to produce a new file. PATCH does not modify the original binary file.

5.2 Commands

Table 5–1 lists the PATCH commands and their functions.

Table 5–1: PATCH Commands

Command	Functions
BOOT	Boot specified device
CLEAR	Clear input table
EXIT	Return to XXDP monitor
GETM	Load DEC/X11 MAP file
GETP	Load saved input table
KILL	Delete address from input table
MOD	Enter address in input table
PATCH	Create patched file
SAVP	Save input table
TYPE	Print/display input table on terminal

Section 5.2.1 through Section 5.2.10 describe the PATCH commands.

5.2.1 BOOT Command

You use the BOOT command to boot the specified device.

The format of the command is:

BOOT [dev:]

where:

- **dev:** is the optional name of the device you want to boot. If you omit this parameter, PATCH boots the system device.

5.2.2 CLEAR Command

You use the CLEAR command to clear the input table of all entries.

The format of the command is:

CLEAR

5.2.3 EXIT Command

You use the EXIT command to return control to the XXDP monitor.

The format of the command is:

EXIT

5.2.4 GETM Command

NOTE: *Use this command only when you are patching DEC/X11 RTE files.*

You use the GETM command to retrieve a DEC/X11 MAP file from a specified device and load it into memory. (Section 5.3 discusses DEC/X11 MAP files).

The format of the command is:

GETM [dev:]filnam.ext

where:

- **dev:** is the optional name of the device from which you want to retrieve the file. If you omit this parameter, PATCH searches the system device.
- **filnam.ext** are the name and extension of the DEC/X11 MAP file that you want to retrieve.

5.2.5 GETP Command

You use the GETP command to write the contents of a file created with the SAVP command (see Section 5.2.9) to the input table.

CAUTION: *This command deletes previous input table contents.*

The format of the command is:

GETP [dev:]filnam.ext

where:

- **dev:** is the optional name of the device on which the SAVP-created file resides. If you omit this parameter, PATCH write that file to the system device.
- **filenam.ext** are the name and extension of the SAVP file that you want to load into the table.

5.2.6 KILL Command

You use the KILL command to delete an entry from the input table.

The format of the command is:

KILL addr

where:

- **addr** is the address (octal) of the entry.

5.2.7 MOD Command

You use the MOD command to enter an address and its contents into the input table. You use this command differently to modify a DEC/X11 RTE file and a non-DEC/X11 of binary file.

5.2.7.1 Binary (Non-DEC/X11) Mode

You use the following MOD command format when you alter binary files other than DEC/X11 RTEs.

MOD addr

where:

- **addr** is any valid 16-bit address. This value can be any octal number from 0 to 177777. You can leave out leading zeros.

After you press Return to enter the command, PATCH prints/displays the requested address followed by a slash.

- If the address has not yet been entered in the input table, PATCH prints/displays six dashes after the slash.
- If the address has already been entered in the input table, PATCH prints/displays the previously entered contents after the slash.

You can accept, create, or change the contents:

- Press Return only to close the table entry without creating or changing its contents.

- Enter the new value (octal) and press either Return or Line Feed to close the table entry.
  - If you press Return, PATCH closes the table entry and returns to its prompt.
  - If you press Line Feed, PATCH closes the table entry and prints/displays the next addressable memory location (<ADDR>+2).

The following are examples of using the MOD command.

❶
```
*MOD 123456
123456/------
```

❷
```
*MOD 123456
123456/------000207 Return
*
```

❸
```
*MOD 11040
011040/000240
```

❹
```
*MOD 11040
011040/000240 000137 Line Feed
011042/------ 051502 Return
```

❶ The operator has specified physical address 123456 to be modified. The dashes indicate that this is the first time this address has been specified.

❷ The operator deposits 000207 in physical address 123456. The carriage return closes the input table entry, and PATCH prompts you for a new command.

❸ The operator wants to examine physical address 11040. PATCH responds by printing an actual value after the slash. This indicates that someone has already entered this address into the input table and specified 000240 as the contents.

You can now enter the value you wish to deposit in this address.

❹ The operator examines the table entry for address 11040, changes its contents to 000137, and presses Line Feed to make a table entry for physical address 11042. The dashes indicate that this address has not been previously included in the table. The operator deposits the contents 51502 and presses Return to close the table entry and cause PATCH to prompt for a new command.

5.2.7.2 Modifying a DEC/X11 File

The command has three different formats:

1. **MOD addr**
2. **MOD MON modnam addr**
3. **MOD opmod addr**

where:

- **addr** is either the physical address (octal) whose contents you want to change or offset (octal) into the module of the location whose contents you want PATCH to display.
- **MON** is a keyword that indicates that the module resides in the monitor section of the RTE.
- **modnam** is the name of the module in which the address whose contents you want to change is located.
- **opmod** is the name of the option module in which the address whose contents you want to change is located.

Use format 1 of the MOD command as you would in non-DEC/X11 usage (see Section 5.2.7.1). This command accepts 18-bit addresses.

Use formats 2 and 3 of the MOD command when you want to specify offsets into a DEC/X11 RTE file.

NOTE: *You can use formats 2 and 3 only if you have used the GETM command to retrieve a DEC/X11 MAP file. (See Section 5.3.)*

Use format 2 to modify locations within monitor modules. The following is an example of using format 2:

```
*MOD MON KTERR 24 Return
```

The operator has specified location 24 relative to the beginning of the monitor module named KTERR.

Use format 3 to modify locations within option modules of exercisers. The name of the option module to be modified (**opmod**) has five characters, the fifth character being the copy number (the first copy is 0). The following is an example of using format 3:

```
*MOD CPBJ0 100 Return
```

The operator indicates that the location to be changed is location 100 relative to the beginning of the first copy of option module CPBJ.

The following is an example of using the MOD command to patch a DEC/X11 file:

```
❶ *MOD MON KTERR 10 Return
❷ 007126/000240 137 Line Feed
❸ 007130/------ 51502 Return
```

❶ The operator asks to see the contents of relative location 10 in the module KTERR, which resides in the RTE monitor.

❷ PATCH prints/displays the physical address (007126) and its contents (240). The operator deposits a new value (137) in this address and presses Line Feed to examine the contents of the next addressable location (007130).

❸ The contents have not been entered; the operator deposits 51502 and presses Return to close the table entry.

5.2.8 PATCH Command

You use the PATCH command to produce a new output file after the address modifications have been entered in the input table.

The format of the command is:

PATCH [dev:]ofilnm.ext=[dev:]ifilnm.ext

where:

- **dev:** is the optional name of the output and input device. If you omit this parameter, PATCH searches the system device.
- **ofilnm.ext** are the name and extension of the new output file containing the address modifications contained in **ifilnm.ext**.
- **ifilnm.ext** are the name and extension of the input file containing the address modifications you have made to patch the diagnostic.

PATCH reads the input file, adds the address modifications contained in the input table, and builds a new output file having the specified file name. The execution of this command does not affect the input file and table.

The following is an example of using the PATCH command:

```
*PATCH DY1:SAMPL2.BIN=DY0:SAMPL1.BIN Return
```

This command takes the file SAMPL1.BIN located on device DY0:, combines it with the address modifications in the input table, and produces a new file on device DY1: called SAMPL2.BIN.

Once you press Return to conclude the command, PATCH prints/displays the following instruction:

```
IF THIS IS DECX11 TYPE THE MONITOR TYPE.  ELSE JUST Return
```

If you are patching a file that is *not* a DEC/X11 RTE, press Return. PATCH responds by constructing the output file, printing/displaying the message DONE, and prompting you for a new command.

If you are patching a DEC/X11 RTE file, enter the type of monitor contained in the RTE. The system responds according to whether or not the monitor supports memory management.

- If the monitor does *not* support memory management, PATCH builds the new output file and prints/displays the message DONE on the terminal. PATCH prompts for a new command.
- If the monitor supports memory management, PATCH checks to see if a MAP file has been loaded into memory with the GETM command (see Section 5.2.4).
  - If there is no MAP file in memory, PATCH prompts you for the MODQ address:

    ```
    TYPE MODQ ADDRESS:
    ```

    Enter the MODQ address and press Return. PATCH displays the following message:

    ```
    IF MODIFYING OPTION MODULES, TYPE LOWEST MODULE ADDRESS, ELSE
    JUST Return
    ```

    (The end of this section discusses how to find the MODQ address.)
  - If a MAP file already resides in memory, PATCH displays the following message.

    ```
    IF MODIFYING OPTION MODULES, TYPE LOWEST MODULE ADDRESS, ELSE
    JUST Return
    ```

    (The end of this section discusses how to find the lowest option module address.)
- If you are modifying the monitor section of the RTE, press Return.
- If you are modifying only option modules or if you are patching both the monitor area and one or more option modules, enter the address of the first option module that was linked into the RTE and press Return.

The PATCH utility constructs the output file, prints/displays the message DONE, and prints/displays its command prompt.

Use one of these methods to find the RTE monitor type.

- Run the DEC/X11 configurator/linker program and enter the configuration file for this RTE, if it exists.
- Run the DEC/X11 configurator/linker program and enter the MAP file for this RTE, if it exists. The monitor type appears at the top of the listing.
- Run the RTE. The monitor type is printed at start-up time.

Chapter 8 gives full information on running these programs.

Refer to an appropriate MAP file (see Section 5.3) to determine the MODQ address.

The symbol MODQ is located within the monitor module CONFIG. Find the module name CONFIG on the listing and look at the symbol names underneath it until you find MODQ. The physical address printed next to the symbol MODQ is the address you enter.

Use one of these methods to obtain the address of the first option module that was linked into the RTE you are patching.

- If you have a MAP file listing for the proper monitor type (see Section 5.3), find the first occurrence of an option module name on the listing and use the physical address associated with that option module.

  Option module names are located at the end of the listing. The physical address of each module is printed next to the module name, under the heading PH ADDR.
- Load and run the RTE that you intend to patch and enter a MAP command before you run the PATCH program. This command prints the addresses of all option modules. Look for the address labelled PA: for each option module. Find the physical address that is lowest. The lowest address may not be the first one printed/displayed.

5.2.9 SAVP Command

You use the SAVP command to save the contents of the input table as a file.

The format of the command is:

SAVP [dev:]filnam.ext

where:

- **[dev:]** is the optional name of the device on which the newly created file resides. If you omit this parameter, the file is saved on the system device.
- **filnam.ext** are the name and extension of the file in which the input table contents are saved.

PATCH writes a file with that name on the device. The command does not alter input table contents.

5.2.10 TYPE Command

You use the TYPE command to print/display the input table contents on the console terminal.

The format of the command is:

TYPE

5.3 DEC/X11 MAP Files

When you modify DEC/X11 RTEs, you must have the MAP file to use the full capabilities of the MOD command. The MAP file contains necessary information about RTE contents, for example, module names, revision levels, and so on.

If you do not use a MAP file, the MOD command accepts only physical addresses as arguments, and you must manually calculate the physical address of the option module's relative address and enter that value as an argument.

On the other hand, if you have used the GETM command to fetch a MAP file, the MOD command accepts as arguments monitor and option module names and offsets into them.

The DEC/X11 configurator/linker produces the MAP file. It is the symbol table that is generated at link time and saved using the SAVM command. (Chapter 8 discusses MAP files.)

While the number and order of the option modules in any RTE is unique, the monitor modules of any given monitor type (A, B, C, and so on) are always linked in the same sequence. For this reason, you need not have the MAP file for the particular RTE you are modifying. If you want to specify the monitor module name plus an offset to the MOD command, you can use any MAP file for the proper monitor type. For example, if you are modifying the monitor area of an RTE of a type C monitor, you can use any

MAP file that was generated when any RTE having a type C monitor was linked.

Similarly, the address of MODQ is also the same for every monitor of the same type. When the PATCH command prompts you for the address of MODQ, you can look at any map listing of the proper monitor type to obtain the address.

NOTE: *All MAP files generated from the previous release of the DEC/X11 monitor library become invalid when a new release of the library is issued. You must produce new MAP files to patch files generated with the new library.*

5.4 Suggested DEC/X11 Application

When a diagnostic engineering patch order (DEPO) is issued for a DEC/X11 monitor module, the patch must be added to every runtime exerciser that is generated containing that module (depending on the monitor type). If you build many RTEs (for example, in a manufacturing environment), you save time and resources by building and saving an input table for every monitor type. You can add information to these tables and resave them every time a new DEPO is issued.

After you build an RTE with the configurator/linker, run the PATCH utility, get the input table for the proper monitor type, and add modifications that must be made to the option modules (using the MAP file if you wish). Then execute the PATCH command.

See Chapter 8 for a full discussion of the DXCL utility.

Chapter 6 SETUP Utility

You use the SETUP utility to build a diagnostic's hardware and software tables before you run the diagnostic. You store these tables with the diagnostic. SETUP has the same memory requirements as DRS: 5.75 Kwords in the lower 28 Kwords of memory. The minimum-size system that can be used is 28 Kwords.

6.1 Starting SETUP

You enter the following to start SETUP:

.R SETUP

SETUP prints/displays an asterisk (*) to prompt you for commands.

6.2 Commands

Table 6–1 lists the SETUP commands:

Table 6–1: SETUP Commands

Command	Function
EXIT	Return control to XXDP
LIST	Print/display a list of DRS diagnostics on a media
SETUP	Build tables for specified diagnostic

6.2.1 EXIT Command

You use the EXIT command to return to the monitor.

The format of the command is:

EXIT.

6.2.2 LIST Command

You use the LIST command to obtain a list of all DRS-compatible diagnostics on a media.

The format of the command is:

LIST [dev:][file.ext]

where:

- **dev:** is the optional name of the device to search for DRS-compatible files. If you omit this parameter, SETUP searches the system device.
- **filnam.ext** is the name of the file(s) to search. The file must have a .BIN or .BIC extension. You can specify wildcards. The default is .BI?

6.2.3 SETUP Command

You use the SETUP command to load the diagnostic into memory and build its tables.

The format of the command is:

SETUP [devo:]ofile=[devi:]ifile

where:

- **devo:** is the optional name of the device to which the new file is to be written. If you omit this parameter, SETUP writes to the system device. The device must be on-line.
- **ofile** is the name of the diagnostic file that receives the new table. The file must have a .BIN or .BIC extension.
- **devi:** is the optional name of the device from which the file containing new table information is to be read. If you omit this parameter, SETUP reads from the system device. The device must be on-line.
- **ifile** is the name of the diagnostic file containing the new table information. The file must have a .BIC or .BIN extension.

SETUP responds by building the diagnostic's tables, prompting you for hardware and software parameters as it does. You answer the prompts as you would if you had issued the START command and were actually running the diagnostic.

You can give the output file the same name as the input file. If you do, SETUP prints/displays this message to warn against the accidental loss of the original file:

```
DELETE ifile?  (Y/N/CR=Y)
```

If you type a Y or N answer, the input file “ifile” is deleted after the SETUP process and the new file is then written.

Chapter 7
XTECO Utility

You use the XTECO (pronounced “ex-TEE-co”) utility to create and modify (edit) text files. XTECO, which is a limited subset of the TECO character editor supported by most of DIGITAL’s operating systems, provides you with necessary commands to edit XXDP text files.

Text files contain ASCII data representing valid text. Valid text consists of all printing characters, tab, carriage return (<CR>), line feed (<LF>) and form feed (<FF>).

The principal text files in XXDP are batch control (chain) files. Chapter 9 discusses setting up and running this type of file.

7.1 Description

Here are the XTECO concepts that you need in order to use the commands:

- To XTECO, a text file contains one long string of characters, like beads on a string. XTECO processes the file one character at a time.
- XTECO can have only a certain number of characters in memory at one time, so you can work with only a segment of the string at once.
- XTECO processes files by reading a segment from the input file into memory, editing that segment as you direct, and writing it to the output file. This process continues until the entire input file has been acted upon.
- In the case of the TECO command (see Section 7.1.1), where the input and output file may have the same name and reside on the same media, XTECO creates a temporary output file which replaces the input file after the edit process is complete.
- Special characters (<CR> and <LF>) act as signals to XTECO, telling it when one line of text ends and another begins. XTECO can manipulate lines of text as well as characters.
- The editor displays the cursor to show you where it is located on the string. You issue various commands to move the cursor around and

indicate to XTECO where old text is to be removed or modified and new text is to be placed.

You can move the cursor over only the portion of the string currently in memory. You cannot move the cursor backwards into the portion of the string that has already been processed and placed into the new file.

7.1.1 Starting the Edit Process

To start the XTECO utility, type:

.R XTECO

The system displays an asterisk (*) to prompt you to begin an editing session.

Start the text editing process by issuing one of these commands:

1. **EDIT [devo:]ofile=[devi:]ifile**
2. **TECO [devi:]ifile**
3. **TEXT [devo:]ofile OR [devi:]ifile**

where:

- **devo:** is the name of the output device on which the new file is to be stored.
- **ofile** is the name of the file into which the output is to be written.
- **devi:** is the name of the input device from which the old file is to read.
- **ifile** is the name of the input file.

The system responds by putting XTECO into edit mode, displaying a quotation mark (") to prompt you for commands, and displaying the text in the XTECO buffer (approximately 30 lines).

You can now create, change, or delete text.

You select one of the above commands as follows:

- Use the EDIT command to edit an already existing text file. You use the EDIT command with any type of device; however, the input and output devices must be of the same type, for example, DU: and DU:.
- Use the TECO command only when you edit an existing text file that resides on a random-access device, such as a disk.

  You need type only the name of the file you want to edit (the input file). XTECO automatically creates an output file resulting from your edits and gives it the same name and extension as the input file. You can choose to have XTECO replace the input file with the output file at the end of your editing session.

- Use the TEXT command to create a new text file.

7.1.2 Command Summary

Table 7–1 lists the commands available in edit mode by type of function.

Table 7–1: XTECO Command Summary

Command	Function
Cursor Location	
C	Move the cursor character by character
J	Move the cursor to the beginning of text
L	Move the cursor line by line
ZJ	Move the cursor to the end of text
Search	
N	Search for specified string in remainder of text file
S	Search for specified string in text now in memory
Modify/Display Text	
A	Append text to that currently in memory
D	Delete character(s)
K	Delete line(s)
I	Insert text
T	Type text
Terminate Edit Mode	
EX	Exit edit mode

You terminate all commands by typing two altmode (<ESCAPE>) characters. The altmode key is usually the left uppermost key on DIGITAL terminals. The terminal echoes the altmode character as a $.

7.2 Commands

NOTE: *In the following examples, the caret (^) shows the position of the cursor. Do not enter this character in actual operation. The dollar sign ($) represents the altmode character.*

The following sections explain each edit-mode command and give examples of its use. Section 7.5 gives sample edit sessions.

7.2.1 A Command

You use the A command to append the next section of text residing in the input file to the text that you are editing, which is already located in memory.

The format of the command is:

A$$

The combined sections of text in memory are now treated as a single section. You may append as many sections of input-file text as memory size limits allow.

7.2.2 C Command

You use the C command to move the cursor on a character-by-character basis, either forward or backward any number of characters within the text currently located in memory.

The format of the command is:

[n]C$$

where:

- **n** is an optional argument that specifies the direction and the number of characters to move. It is a decimal number that is positive for forward motion and negative for backward motion. If you omit this number, XTECO moves the cursor forward one character.

This command does not recognize lines; XTECO treats the <CR> <LF> sequence as two characters.

The following are examples of using the C command.

Command	Text in Memory
–	;NEXT COMMAND WILL T^EST RP06 ;ALL ERRORS WILL BE REPORTED
"C$$	;NEXT COMMAND WILL TE^ST RP06 ;ALL ERRORS WILL BE REPORTED
"-3C$$	;NEXT COMMAND WILL^ TEST RP06 ;ALL ERRORS WILL BE REPORTED
"10C$$	;NEXT COMMAND WILL TEST RP06^ ;ALL ERRORS WILL BE REPORTED
"3C$$	;NEXT COMMAND WILL TEST RP06 ;^ALL ERRORS WILL BE REPORTED

In the last command, the <CR> <LF> sequence counts as two characters.

7.2.3 D Command

You use the D command to delete characters from the text in memory.

The format of the command is:

[n]D$$

where:

- **n** is an optional argument that specifies the number of characters to be deleted. It is a decimal number that is positive if the characters follow the current cursor position and negative if they precede it. If you omit this number, XTECO deletes 1 character forward.

You can delete any number of characters preceding or following the current cursor position.

Following are examples of the D command.

Command	Text in Memory
–	;COM^MENT LINE IN BATCH CONTROL FILE
"4D$$	;COM^ LINE IN BATCH CONTROL FILE
"-3D$$	;^ LINE IN BATCH CONTROL FILE
"D$$	;^LINE IN BATCH CONTROL FILE

7.2.4 EX Command

You use the EX command after you have completed editing a text. The command closes the output file and writes it back to the device.

The format of the command is:

EX$$

XTECO leaves edit mode and displays an asterisk (*) to prompt you to begin another editing session or exit from the utility. (See Section 7.3.1.)

If you use the TECO command to start the editing session, XTECO renames the input file by giving it a .BAK extension, gives the edited file the same name as the original input file, and stores it on the original device.

7.2.5 I Command

You use the I command to insert new text.

The format of the command is:

Itext$$

where:

- **text** is the text to be inserted.

XTECO inserts the text after the current cursor position and positions the cursor after the new text.

The inserted text can consist of any valid text characters.

The following are examples of using the I command. Line terminators are depicted in these examples. The return typed by the user is represented as [Return]. The two-character sequence generated by the return and stored in the text is represented as <CR><LF>.

Command	Text in Memory
–	R UPDAT<CR><LF> ^
"IPIP[Return] _	R UPDAT<CR><LF> PIP^
"I DU0:=DU1:[Return] $$	R UPD2<CR><LF> PIP DU0:=DU1:<CR><LF> ^
–	FILE DU1:=DU^:<CR><LF>
"I2[Return] _	FILE DU1:=DU2^:<CR><LF>

7.2.6 J Command

You use the J command to position the cursor at the beginning of all text currently located in memory.

The format of the command is:

J$$

Following is an example of the J command:

Initial state:

```
R PROG1
R PROG2
R PROG^3
```

After execution of the J command:

```
^R PROG1
R PROG2
R PROG3
```

7.2.7 K Command

You use the K command to delete lines of text from the text located in memory.

The format of the command is:

[n]K$$

where:

- **n** is an optional argument that specifies the number (decimal) of lines to be deleted. It is a decimal number that is positive if the line(s) follows the current cursor position and negative if they precede it. If you omit this number, XTECO deletes the line in front of the cursor position, including the <CR> <LF> sequence.

A line of text is a string of characters between two <CR> <LF> sequences. You produce this sequence when you press RETURN.

The following are examples of using the K command.

Command	Text in Memory
–	;START OF CONTROL FILE ^R PROG1 R PROG2 R PROG3 R PROG4 ;END OF FILE
"K$$	;START OF CONTROL FILE ^R PROG2 R PROG3 R PROG4 ;END OF FILE
"-1K$$	^R PROG2 R PROG3 R PROG4 ;END OF FILE
"2K$$	^R PROG4 ;END OF FILE
"3C$$	R P^ROG4 ;END OF FILE
"K$$	R P^;END OF FILE

Note the effect of the K command when the cursor is not positioned at the beginning of a line. You can easily determine the effect of any K command by issuing a T command with the identical format. Whatever is typed after you issue the T command is what will be deleted by the K command. (The commands 3T and 3K are of identical format.)

7.2.8 L Command

You use the L command to move the cursor on a line-by-line basis. You can move the cursor backward or forward any number of lines of the text currently in memory.

The format of the command is:

[n]L$$

where:

- **n** is an optional argument that specifies the direction to move the cursor and the number of lines to move it. It is a decimal number that is positive if the line(s) follows the current cursor position and negative if they precede it. If you omit this number, XTECO moves the cursor one line forward.

XTECO responds by moving the cursor and positioning it at the beginning of a line.

A line of text is a string of characters between two <CR> <LF> sequences. You produce this sequence when you press RETURN.

The following are examples of using the L command.

Command	Text in Memory

–	R ZRLG^??
	IF ERROR THEN
	PRINT RL02 HAS HARDWARE PROBLEM
	END

"L$$	R ZRLG?? ^IF ERROR THEN PRINT RL02 HAS HARDWARE PROBLEM END

"-1L$$	^R ZRLG??
	IF ERROR THEN
	PRINT RL02 HAS HARDWARE PROBLEM
	END

"2L$$	R ZRLG??
	IF ERROR THEN
	^PRINT RL02 HAS HARDWARE PROBLEM
	END

7.2.9 N Command

You use the N command to search for a specified string of characters in the text located in memory and in the remaining stored text.

The format of the command is:

Nstring$$

where:

- **string** is the character sequence to search for. This string can consist of any number of valid characters. including tab, <CR>, and <LF>.

XTECO begins at the current cursor position and searches through the text in memory. If XTECO does not find a match of the search string in memory, it writes the text it has already examined to the output file, and reads more text into memory from the input file.

XTECO stops when it matches the search string or finds the end of text.

The N command has the effect of a nonstop S command. (See Section 7.2.10.)

You cannot use the J command (see Section 7.2.6) to go back to the beginning of a file. XTECO has already written the section of text that you searched to the output file. You must exit from edit mode and reenter it to edit the output file.

7.2.10 S Command

You use the S command to search for a specified string of characters in the text currently located in memory.

The format of the command is:

Sstring$$

where:

string is the character sequence to search for. This string can consist of any number of valid characters.

XTECO starts at the cursor position and searches forward through the text in memory. XTECO stops as soon as it has matched the search string or come to the end of text in memory.

If XTECO finds the string, it positions the cursor after it. If XTECO does not match the search string, it prints/displays a message to that effect and positions the cursor at the end of the text in memory. You can use the J command (see Section 7.2.6) to place the cursor back at the beginning of text in memory.

If you want to search all of the remaining text, use the N command. (See Section 7.2.9.) The following are examples of using the S command.

Command	Text in Memory
_	^R UPDAT PIP DU0:=DU2:*.BIN EXIT
_"SDU1$$	R UPDAT PIP DU0:=DU2:*.BIN EXIT^
_"J$$	^R UPDAT PIP DU0:=DU2:*.BIN EXIT

Command	Text in Memory
"SDU2$$	R UPDAT PIP DU0:=DU2^:*.BIN EXIT

In the above example, the first search fails, and XTECO prints/displays the error message:

```
NOT FOUND:  DU1
```

7.2.11 T Command

You use the T command to print/display text on the console terminal.

The format of the command is:

[n]T$$

where:

- **n** is an optional argument that specifies the number of lines to be typed. It is a decimal number that is positive if the lines follow the current cursor position and negative if they precede it. If you omit this number, XTECO types the text after the cursor on the current line. If the cursor is positioned within a line of text, typing starts/concludes at the cursor position (see examples).

A line of text is a string of characters between two <CR><LF> sequences. You produce this sequence when you press RETURN.

HT is a special form of the T command that causes all text currently located in memory to be typed, regardless of the position of the cursor.

The following are examples of using the T command.

The initial state of the cursor is as follows:

```
;BATCH CONTROL FILE FOR TESTING THE DZ11
^R ZDZA??
R ZDZB??
;END OF DZ11 TESTING
```

Command	Text Typed
"T$$	R ZDZA??
"1T$$	;BATCH CONTROL FILE FOR TESTING THE DZ11

Command	Text Typed
"2T$$	R ZDZA?? R ZDZB??

The initial state of the cursor is as follows:

```
R PROG1
R PRO^G2
R PROG3
```

Command	Text Typed
"T$$	G2
"-1T$$	R PROG1 R PRO
"0T$$	R PRO
"HT$$	R PROG1 R PROG2 R PROG3

7.2.12 ZJ Command

You use the ZJ command to position the cursor after all text currently in memory.

The format of the command is:

ZJ$$

The following is an example of using the ZJ command:

Initial state of text:

```
^R PROG1
R PROG2
R PROG3
```

Text after issuing the ZJ command:

```
R PROG1
R PROG2
R PROG3^
```

7.3 Nonedit Commands

You use three XTECO nonedit commands to perform editing-related functions or exit from the XTECO utility.

7.3.1 EXIT Command

You use the EXIT command to return control to the XXDP monitor.

7.3.2 PRINT and TYPE Commands

You use the PRINT and TYPE commands to print/display text files on the line printer and console terminal respectively. They are equivalent to the UPDAT commands of the same name. See Section 4.1.2.4.

7.4 Combining Edit Commands

You can combine several edit mode commands on a single command line. You do this by separating each command with a single altmode character and then terminating the entire string of commands with two escapes. For example, the following commands:

```
"NTEST$$
"OT$$
"T$$
```

can be combined into a single string:

```
"NTEST$OT$T$$
```

In both cases, XTECO will do a nonstop search for the string TEST, type the characters from the beginning of the line where the string was found to the current cursor position, and then type the characters from the current cursor position to the end of the line.

Combining commands speeds up your job, but DIGITAL suggests that you do not use this feature until you are familiar with the operation of individual commands.

7.5 Sample Edit Sessions

Here are two sample edit sessions that show you the various ways of handling XTECO. In these examples, XTECO types the underlined text. Brackets enclose the comments that accompany the examples. The comments do not represent actual dialogue between the user and the system.

7.5.1 Simple Method for Creating a Text File

```
.R XTECO
TEXT TEST.CCC
[User creating new file on
system device called TEST.CCC]
"I;THIS IS A BATCH CONTROL JOB FOR TESTING THE RX01
;THIS IS A FICTIONAL JOB FOR DEMO OF XTECO ONLY!
;
R ZRXX??
RES/PAS:1
N
EXIT
;THE FIRST RX01 DIAGNOSTIC HAS BEEN RUN.
;
R ZRXY??
RES/PAS:1/TES:1-5
N
EXIT
;END OF RX01 TEST
$$
"EX$$
EXIT
.
```

7.5.2 Changing an Existing Text File

```
.R XTECO
TECO TEST.CCC
[The user is going to change TEST.CCC,
on the system device.]
"SRX02$$
? NOT FOUND: RX02
"J$$
""T$$
;THIS IS A BATCH CONTROL JOB FOR TESTING THE RX01
"L$$
"2T$$
;THIS IS A FICTIONAL JOB FOR DEMO OF XTECO ONLY!
;
"EX$$
EXIT
```

7.5.3 Use of Combined Edit Commands

```
.R XTECO
EDIT DL1:TEST1.CCC=DL1:TEST.CCC

[EDIT TEST.CCC, WHICH IS ON THE SYSTEM
DEVICE, AND PLACE THE EDITED OUTPUT
INTO A FILE CALLED TEST1.CCC ON DL1.]

"NPAS:$0TT$$ [SEARCH FOR "PAS:" AND TYPE LINE]
RES/PAS:1
"DI2$0TT$$ [DELETE NEXT CHAR AND INSERT "2"]
RES/PAS:2
"EX$$
EXIT

.
```

Chapter 8

DXCL (DEC/X11 System Exerciser Utility)

You use the DEC/X11 system exerciser utility (DXCL) after you have tested individual units to create and run independent runtime exerciser (RTE) programs that drive associated systems at maximum activity rates in order to test their reliability.

You customize each RTE to suit the hardware and configuration of the PDP-11 system you want to test by choosing from among available DEC/X11 components. You run each RTE using the commands listed in Section A.9.

Section 8.1 discusses selecting DEC/X11 RTE components.

Section 8.2 discusses configuring and linking your RTE.

8.1 Selecting DEC/X11 RTE Components

DEC/X11 RTE components consist of the following programs:

- A DEC/X11 monitor, chosen from among 11 available monitors
- Device/option modules, each of which is dedicated to testing a single option or device controller
- A configurator/linker program corresponding to the monitor, which enables you to combine and link the monitor and the selected modules into a customized RTE

8.1.1 DEC/X11 Monitors

There are 11 available DXCL monitors. Each is designed for a particular memory configuration, processor choice, and bus design. You choose from among them according to the hardware configuration and the options of the system under test. Hardware configuration and options include the following:

- PDP-11 CPU type (for example, 11/84 and microPDP)

- Available memory size in these ranges: up to 28K words, up to 124K words, and up to 2048K words
- Available memory options (for example, memory management [KT], cache memory, memory parity checking, ECC)
- Available devices and options (for example, line printer, line clock, and so on)
- Special operations (for example, error logging, 22-bit addressing, and so on)

Table 8–1 lists the monitors and types of devices to which they correspond.

Table 8–1: Available Monitors

Monitor Type	Device Type
A,B	Non-KT (memory management) single-processor monitors (up to 28K words)
C,D	Single-processor monitors with KT (up to 124K words)
E	Single-processor monitors with KT (up to 2048K words)
Q	22 bit LSI bus structure (up to 2048K words)
F	Non-KT APT single-processor monitors (up to 28K words)
G,H	APT single-processor monitors with KT (up to 124K words)
I	APT single-processor monitors with KT (up to 2048K words)
R	APT 22 bit LSI bus structure (up to 2048K words)

Table 8–2 lists the monitors and the functions each is capable of testing.

Table 8–2: DEC/X11 Monitor Capabilities Summary

Commands and Capabilities	A	B(F)	C(G)	D(H)	E(I)	Q(R)
11/44 error logging					X	
11/60 error logging				X		
11/70 error logging					X	
22-bit addressing					X	X
22ON (22-bit addressing on)					X	X

Table 8–2 (Cont.): DEC/X11 Monitor Capabilities Summary

Commands and Capabilities	A	B(F)	C(G)	D(H)	E(I)	Q(R)
22OFF (22-bit addressing off)					X	X
Bad vector service			X	X	X	X
COFF (cache memory off)			X	X	X	
CON (cache memory on)			X	X	X	
DES (deselect module)	X	X	X	X	X	X
EXAM (examine location)		X	X	X	X	X
EXIT (exit to monitor)			X	X	X	X
FILL (fill char./count)	X	X	X	X	X	X
LPOFF (line printer off)		X	X	X	X	X
LPON (line printer on)		X	X	X	X	X
Memory management (KT)			X	X	X	X
MOFF (unibus map off)					X	
MON (unibus map on)					X	
NPR transfers in 28K words	X	X	X	X	X	X
NPR transfers in 124K words			X	X	X	X
NPR transfers in 2048K words					X	X
POFF (parity checking off)		X	X	X	X	X
PON (parity checking on)		X	X	X	X	X
ROTOFF (buffer rotation off)		X	X	X	X	X
ROTON (buffer rotation on)		X	X	X	X	X
RTE relocation			X	X	X	X
RUN (start run mode)	X	X	X	X	X	X
RUNL (start run locked mode)			X	X	X	X
SEL (select module)	X	X	X	X	X	X
SUM (output summary)	X	X	X	X	X	X
SWR (modify switch reg.)	X	X	X	X	X	X
System clock		X	X	X	X	X

8.1.2 Device/Option Modules

Each module is a program that tests a single option or device controller in its interaction with an entire system to find out how the performance of the option or controller relates to a system problem. You choose from among available modules according to the option or device you want to test.

Table F–1 lists devices and options and their corresponding modules.

8.2 Configuring and Linking a DEC/X11 Exerciser

Configuring a DEC/X11 exerciser consists of the following steps:

1. Plan your build (see Section 8.2.1)
2. Run the DEC/X11 configurator/linker to assemble the RTE by constructing its configuration table (C-table) from the information you have gathered (see Section 8.2.2)
3. Use the LINK command to format and create an executable RTE (see Section 8.2.4)

Section F.3 gives a complete sample build.

You may use DXCL I/O commands while you are configuring and linking the RTE to direct the listing, storage, and retrieval of an RTE's C-table and load map. See Section 8.2.6.

8.2.1 Planning the Build

Planning the build consists of the following steps:

1. Note the major hardware components and options of the system you want to test, such as:
   - The processor type and available options (for example, KT, CACHE, and so on)
   - The associated device(s) and available options (for example, DUAL PORTS, and so on)

   Also note the device addresses (DVA), vectors (VCT), priority levels (BR1, BR2), and the counts required to define the numbers of devices (DVC).
2. Use the information to:
   - Determine the types of software (monitor and modules) required to accommodate the hardware configuration to be tested
   - List the monitor type and module names

- List the parameters (default or specified) associated with each module name

Figure 8–1: Sample DEC/X11 System Configuration Worksheet

```
                    DEC/X11 System Configuration Worksheet

Selected DEC/X11 Monitor For Listed CPU and CPU options:  C ❶
                    FILE:  EXERR1.BIN   DATE:  20 SEPT 88

DEVICE           MOD❷R  DVA❸     VCT  BR1 BR2 DVC  SR1   SR2 SR3 SR4

KW11-A           KWA  A  177546*  100* 6*  0*  1*   4
LS11/LV01        LPA  A  177514*  200* 4*  0*  1* 10000
RX11/RX01        RXA  A  177170*  264* 5*  0*  2*
TMB11/TS03       TMA  A  172520*  224* 5*  0*  1*
RP11E/RP02       RPA  A  176710*  254* 5*  0*  2
RK11-D/RK05      RKA  A  177400*  220* 5*  0*  1*
RK611/RK06       RKB  A  177400*  210* 5*  0*  1*
EIS              CPB  A
11/34 Instr.     CPA  A
FP11-A           FPB  A

* =  SOFTWARE DEFAULTS
```

❶ A particular monitor (C) has been derived for a given processor (PDP-11/34) and its options.

❷ Specific test modules have been derived (KWA, LPA, and so on) for both the processor and its associated devices.

❸ Default and specified parameters (DVA, VCT, and so on) are listed.

8.2.2 Assembling the RTE

You use the DEC/X11 configurator/linker to combine the monitor you have chosen with the modules you have chosen to create a binary RTE. Section H.2 lists configurator/linker error messages.

To start the DEC/X11 configurator/linker, type:

.R DXCL

After DXCL has been successfully loaded, it identifies itself and outputs a restart address and a help query, as in the following example:

```
DXCL.BIC

DXCL - XXDP V2 DEC/X11 CNF/LNK REVISION I
RESTART: 006606
DO YOU WANT HELP?(Y <CR> OR JUST <CR>)
```

If you press Return, DXCL prints its help list, which contains all DXCL keyboard commands (see Table A–10). If you press N and Return, DXCL displays an asterisk (*) to prompt you for commands.

8.2.3 Configure Mode Commands

You can now enter configure mode (see Section 8.2.3.3) and use DXCL's configure mode commands to build your RTE's configuration table (C-table). The table contains the monitor name and one entry for each module to be included in the RTE. Each module entry in turn contains nine parameters that you fill in:

1. Device address (DVA)
2. Vector address (VCT)
3. Bus request level 1 (BR1)
4. Bus request level 2 (BR2)
5. Device count (DVC)
6. Software switch register 1 (SR1)
7. Software switch register 2 (SR2)
8. Software switch register 3 (SR3)
9. Software switch register 4 (SR4)

DXCL uses the C-table's contents at link time to assemble the selected monitor and modules.

Figure 8–2 shows a sample C-table. Table 8–3 lists DXCL configure mode commands.

Figure 8–2: Sample Configuration Table

```
*GETC TEXT.CNF      ;Retrieve configuration table
DONE
*TYPEC              ;Display configuration table

Q
CPA     DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
          SR1-000000 SR2-000000 SR3-000000 SR4-000000
CPB     DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
          SR1-000000 SR2-000000 SR3-000000 SR4-000000
KWA     DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
          SR1-000000 SR2-000000 SR3-000000 SR4-000000
FPA     DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
          SR1-000000 SR2-000000 SR3-000000 SR4-000000
DHA     DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
          SR1-000000 SR2-000000 SR3-000000 SR4-000000
KXA     DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
          SR1-000000 SR2-000000 SR3-000000 SR4-000000
DUB     DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
          SR1-000000 SR2-000000 SR3-000000 SR4-000000
```

Table 8–3: Configure Mode Commands

Command	Function
BR1, BR2 number	Enter priority levels
CL	Clear C-table
CNF	Initiate configure mode
DVA addr	Enter device address
DVC number	Enter device count
EX	Exit configure mode
KI	Delete current entry
MDL modnam	Enter module name or output module entry
MON modnam	Enter monitor name
NXT	Output next module entry
POINT modnam	Output specified module entry
VCT addr	Enter vector address
SR1-SR4 number	Enter values in software switch registers

8.2.3.1 BR1 and BR2 Commands

You use the BR1 and BR2 commands to enter the bus request (BR) levels of the current module entry.

The format of the command is:

BRn Brnumber

where:

- **n** is either 1 or 2, a number identifying the first or second bus request level in the module entry
- **Brnumber** is an octal number from 0–7 indicating the bus request level

The system responds by entering the bus request level in the high-order (Brl) or low-order (Br2) byte of the word specifying the bus request level in the current module entry.

You typically use the BR1 and BR2 commands when the device/option's DEC/X11 module does not provide default levels. Consult the listing if you are unsure. You also use the DVA command when the BR levels are nonstandard in relation to the device employed. Consult the listing if you are unsure.

The following are examples of using this command:

```
❶ *BR1 6
❷ *BR2 4
❸ *MDL
❹ WXYZ  DVA-177600 VCT-000200 BR1-000300 BR2-000200 DVC-000000
```

❶ Enter bus request level 6 in BR1 of the current module entry

❷ Enter bus request level 4 in BR2 of the current module entry

❸ Output current module entry

❹ The system responds by printing/displaying the contents of the current module entry.

8.2.3.2 CL Command

You use the CL command to clear the entire C-table.

The format of the command is:

CL

DXCL clears the C-table and prompts you for a new monitor name.

The following is an example of using the CL command:

```
❶ *CL
❷ *MONITOR:
```

❶ Clears the entire C-table

❷ The system prompts for a new monitor name.

8.2.3.3 CNF Command

You use the CNF command to enter DXCL configure mode in order to create or change an RTE's C-table.

The format of the command is:

CNF[/NP]

where:

- **/NP** is the optional no-prompt switch.
  - If you omit this switch and then enter an MDL (enter module name) command, DXCL displays nine successive requests for parameters to include in the entries corresponding to the module.
  - If you specify this switch and then enter an MDL command, DXCL does not prompt you for parameters, but you can enter them using the appropriate configure mode commands.

DXCL responds by entering configure mode and prompting you for a monitor name. Once you have entered the monitor name, you may issue an MDL command to create an entry for that module in the C-table and then fill in the entry's parameters.

If you have omitted the /NP switch and DXCL prompts you for module entry parameters, you can inhibit prompting by pressing Ctrl/C. If you do, DXCL stores the parameters you have already entered.

If you have specified the /NP switch and DXCL does not prompt you for module entry parameters, you can reinitiate prompting by issuing the CNF command.

The following are examples of using the CNF command.

Example 8–1: Using the CNF Command without the No-Prompt Switch

```
❶  *CNF [Return]
❷  *MONITOR:  A [Return]
❸  *MDL  WXYZ [Return]
❹  DVA   177540 [Return]
❹  VCT   204 [Return]
❹  BR1   5   [Return]
❹  BR2   5   [Return]
❹  DVC   2 [Return]
❹  SR1-    [Return]
❹  SR2-  4000 [Return]
❹  SR4-    [Return]
❹  SR4-    [Return]

❺  WXYZ DVA-177540 VCT-000230 BR1-000240 BR2-000240 DVC-000003
     SR1-000000 SR2-004000 SR3-000000 SR4-000000
```

❶ Enter configure mode without the /NP switch

❷ The system prompts for a monitor name; enter the monitor name.

❸ Open entry for RTE module WXYZ

❹ The system prompts successively for the entry's parameters.

❺ Once you have filled in the data, the system displays the contents of the module entry.

Example 8–2: Using the CNF Command with the No-Prompt Switch

```
❶  *CNF/NP [Return]
❷  *MONITOR:  B [Return]
❸  *MDL QRST [Return]
❸  *DVA 177540 [Return]
❸  *VCT  240 [Return]
     .
     .
     .
```

❶ Enter configure mode with the /NP switch

❷ The system prompts for a monitor name; enter the monitor name.

❸ Open entry for module QRST

❹ Manually enter the device address (DVA) and the vector (VCT) into the module entry

8.2.3.4 DVA Command

You use the DVA command to enter a device address parameter into the current module entry.

The format of the command is:

DVA address

where:

- **address** is the 6-digit octal address of the device. You can enter only an even-numbered address.

DXCL responds by entering the address in the current module entry.

You typically use the DVA command when the device/option's DEC/X11 module does not provide the default address. (Consult the listing if you are unsure.) You also use the DVA command when the provided address is nonstandard in relation to the actual device employed (for example, a second RP11 disk or TM11 magtape controller).

The following is an example of using the DVA command:

```
❶  *MDL WXYZ [Return]
❷  *DVA 177600 [Return]
❸  *MDL [Return]
❹  WXYZ  DVA-177600 VCT-000200 BR1-000000 BR2-000000 DVC-000000
```

❶ WXYZ is the current module entry.

❷ Enter DVA parameter 177600

❸ Output the current module entry

❹ The summary is incomplete (SR1-SR4 is omitted) but shows that the DVA parameter has been filled.

8.2.3.5 DVC Command

You use the DVC command to define the number of subdevices (for example, drives) or multiple devices (for example, 8DL11s) to be tested by the module.

The format of the command is:

DVC n

where:

- **n** is a decimal integer defining the number of subdevices. You may define up to 16 devices. This number must equal the actual number of subdevices to be consecutively tested.

The system responds by converting the number to its octal equivalent and entering it as the module's DVA parameter. This octal number represents a binary bitmap in which the weight of each bit represents the logical number of each device.

Since multiple devices must be tested consecutively and accessed by consecutive device addresses (whether ascending or descending), you must enter the exact number of devices and make sure that there are no gaps in your addressing scheme.

The following is an example of using the DVC command:

```
❶  *DVC 5 Return
❷  *MDL Return
❸  *WXYZ . . .  DVC-000037 . . .
```

❶ Enter a device count of 5

❷ Output the current module entry

❸ The summary is not complete. However, it shows that the decimal device count 5 has been converted to the octal number 37.

8.2.3.6 EX Command

You use the EX command to exit from configure mode.

The format of the command is:

EX

The system responds by exiting from configure mode but does not exit from the configurator/linker. You use the EXIT command to do this. See Section 8.2.5.4.

You typically use the EX command after you have completed building your RTE's C-table.

You may issue a CNF command to reenter configure mode. DXCL responds by entering configure mode without prompting you for a monitor name, as it would if you were entering the mode for the first time. Instead, DXCL displays the asterisk to prompt you for further commands and points to the first module entry in the C-table.

The following is an example of using the EX command:

```
❶  *EX Return
❷  *CNF Return
❸  *
```

❶ Exit from configure mode

❷ Reenter configure mode

❸ DXCL displays the command prompt.

8.2.3.7 KI Command

You use the KI command to delete the current module entry from the C-Table:

The format of the command is:

KI

The system responds by deleting the last entry referenced (including all its associated parameter values).

When you use the KI command to delete a module entry and then use the TYPEC or PRINTC command to request a summary of the C-table (see Section 8.2.6.7 and Section 8.2.6.3), DXCL displays an empty indicator (<EMPTY>) instead of the deleted module entry.

8.2.3.8 MDL Command

You use the MDL command to create a module entry in the C-table or to output the contents of the current module entry.

The format of the command is:

MDL [name]

where:

- **name** is the optional name of the module you want to include in the C-table.

  If you specify **name**, DXCL enters the module's name in the first available slot in the C-table and opens the module entry so that you can fill it in.

If you omit **name**, DXCL prints/displays a summary of the current module entry.

Name must be the four-character name of an existing module that contains these elements:

- Two characters indicating the device/option name
- One character indicating the specific module; others may exist for the same device/option.
- One-character indicating the module version. You can substitute a ? for this version letter. If you do, DXCL includes the proper version letter at link time.

The following are examples of using the MDL command:

```
❶  *MDL WXYZ [Return]
❷  *MDL [Return]
❸  WXYZ  DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
  SR1-000000 SR2-000000 SR3-000000 SR4-000000
```

❶ Open a module entry for device WXYZ in the current C-table

❷ Output the contents of the current module entry

❸ Module WXYZ is the current entry and its parameters are zeroes.

8.2.3.9 MON Command

You use the MON command to change the C-table's monitor entry.

The format of the command is:

MON monnam

where:

- **monnam** is the name of the monitor that replaces the one already entered in the C-table. Table 8–1 lists available monitors. Table 8–2 summarizes monitor capabilities.

DXCL responds by changing the monitor entry in the C-table.

8.2.3.10 NXT Command

You use the NXT command to output the contents of the next module entry. The next entry is the one following the last-referenced (current) entry.

The format of the command is:

NXT

If a next entry does not exist, DXCL prints/displays an asterisk (*) to prompt you for further commands.

8.2.3.11 POINT Command

You use the POINT command to search the C-table for a specified module entry and to output its contents.

The format of the command is:

POINT name

where:

- **name** is the 4-character name of the module entry you want to find.

DXCL searches the C-table starting at the current or last-referenced entry. If DXCL does not find the entry, it displays the message:

```
? INVALID NAME
```

8.2.3.12 SR1–SR4 Commands

You use the SR1 through SR4 commands to enter octal values into a module entry's four software switch registers (SRs).

The format of the command is:

SRn oooo

where:

- **n** is the number of the SR into which you want to write data.
- **oooo** is an octal number corresponding to the binary number of the bit(s) you wish to set.

A module has four 16-bit SRs provided for general-purpose program switching. You set the bits in SRs to define unique device options, to point to specific module routines, or both. Section 8.2.5.3.5 provides further information on the contents of SR1–SR4.

The following is an example of using the SRn command:

```
❶  *SR2 4000 [Return]
❷  *MDL [Return]
❸  WXYZ  . . .  SR2-004000 . . .
```

❶ Set bit 11 in SR2

❷ Output current module entry

❸ Bit 11 in SR2 has been set. When bit 11 is set, the line printer to be used has 132 columns instead of 80.

8.2.3.13 VCT Command

You use the VCT command to enter the device vector address in the current module entry.

The format of the command is:

VCT addr

where:

- **addr** is the device vector address, an octal number up to 774. The address must be an even number.

You typically use the VCT command when a vector address is unknown and therefore not already provided or when the provided vector is located at a nonstandard address.

The following is an example of using the VCT command:

```
❶  *VCT 200
❷  *MDL

❸  WXYZ  DVA-177600 VCT-000200 BR1-000000 BR2-000000  . . .
```

❶ Enter the vector address of 200

❷ Output the current module entry

❸ Vector address has been entered in C-table.

8.2.4 Linking the RTE

You use the LINK command to produce an executable RTE once you have built the RTE's C-table.

Once you have successfully built an RTE and loaded it onto an output device, you can run it on the system for which it was designed. Section A.9 lists the DXCL commands you use to do so.

The format and requirements of the LINK command vary according to whether you write the RTE to a nondirectory device, such as a magnetic tape, or to a directory device, such as a disk.

Section 8.2.4.1 discusses producing an executable RTE whose destination is a nondirectory device.

Section 8.2.4.2 discusses producing an executable RTE whose destination is a directory device.

8.2.4.1 LINK Command: Nondirectory Devices

CAUTION: *Digital does not recommend linking RTEs to tape, unless you are operating in User Friendly Diagnostic (UFD) mode. Linking RTEs to tape in other operational modes may lead to unpredictable results.*

The format of the command is:

LINK devo:<devi:[/MP][/MLP]

where:

- **devo:** is the name of the nondirectory device to which you want the executable RTE written.
- **devi:** is the name of the device containing the C-table and library.
- **/MP** is an optional switch that causes DXCL to type a copy of the load map on the console. If you specify this switch, DXCL does not print/display RTE's transfer address, low limit, and high limit during the LINK command's first pass, as it does in Example 8–3 and Example 8–4.
- **/MLP** is an optional switch that causes DXCL to output a copy of the load map on the system line printer. If you specify this switch, DXCL does not print/display RTE's transfer address, low limit, and high limit during the LINK command's first pass, as it does in Example 8–3 and Example 8–4.

DXCL prompts you for the size of the target system—that is, the system on which the resultant RTE will run. You respond by entering one of the octal numbers given in the following table:

Table 8–4: Target System Size

Size	Response
4K	20000
8K	40000
12K	60000
16K	100000
20K	120000
24K	140000
28K and greater	160000

The linking program enters the first phase of the linking process. DXCL requests the monitor and test module tapes in the same order as the order defined in the C-table. DXCL performs a partial read of the tapes to ascertain the final structure of the RTE.

The linking program enters the second phase of the linking process. DXCL requests the same tapes in the same order and reads them in their entirety. It extracts the monitor and object modules from the input device as defined by the C-table, produces an executable RTE program, and writes it to the output device.

Once the linking process has successfully completed, DXCL prints/displays a message to that effect, returns to command mode, and prints/displays an asterisk (*) to prompt you for further commands.

The following is an example of using the LINK command to produce an executable RTE and write it to a nondirectory device. Object module tapes are input from the system device, and the completed RTE is written to a magnetic tape (MU0:).

Example 8–3: Linking a Nondirectory Device

```
*LINK MU0:<??:                          ;Link RTE from system device

SYS SIZE:  160000 [Return]              ;RTE memory requirement

MAKE OUTPUT READY. WRITE ENABLE         ;Enable output device

TYPE(CR) WHEN READY [Return]                  ;Acknowledge enable

PASS 1                                  ;Scan all modules

ANY MORE MONITOR TAPES, CASSETTES, ETC.? (YES,NO)
                                        ;Monitor tape request

YES  [Return]                           ;Acknowledge tape

RELOAD INPUT WITH NEXT TAPE, CASSETTE, ETC.
    .
    .
    .
PASS 2                                  ;Link and output RTE module

INPUT TAPES, CASSETTES, ETC. IN SAME SEQUENCE AS IN PASS 1

TYPE(CR) WHEN READY [Return]            ;Acknowledge tape

ANY MORE MONITOR TAPES, CASSETTES, ETC.? (YES,NO)
    .
    .
    .
LINK DONE                               ;Link process completed
```

8.2.4.2 LINK Command: Directory Devices

The format of the command is:

LINK [devo:][filnam.ext]<[devi:]libnam.LIB[/MP][/MLP]

where:

- **devo:** is the optional name of the device to which you want the RTE written. If you omit this parameter, DXCL writes the RTE to the system device.
- **filnam.ext** are the name and the extension of the file that receives the RTE. The extension must be .BIN or .BIC.

If a file of the same name and extension already exists, DXCL prints/displays the message `DELETE OLD? Y` Return `OR JUST` Return.

- If you type Y, DXCL deletes the old output file and executes the LINK command.
- If you type only Return, DXCL does not execute the LINK command. Instead, it prints/displays the message `? USE NEW FILE NAME` and issues an asterisk (*) to prompt you for further commands.

- **devi:** is the optional name of the device on which the configured C-table and the module library reside. If you omit this parameter, DXCL reads the C-table and library file from the system device.
- **libnam.LIB** are the file name and extension of the monitor library against which the RTE is linked. The extension is always .LIB.
- **/MP** is an optional switch that causes DXCL to type a copy of the load map generated by the LINK command on the console. If you specify this switch, DXCL does not print/display RTE's transfer address, low limit, and high limit during the LINK command's first pass, as it does in Example 8–3 and Example 8–4.
- **/MLP** is an optional switch that causes DXCL to output a copy of the load map generated by the link command on the system line printer. If you specify this switch, DXCL does not print/display RTE's transfer address, low limit, and high limit during the LINK command's first pass, as it does in Example 8–3 and Example 8–4.

DXCL prompts you for the size of the target system—that is, the system on which the resultant RTE will run. Respond by entering one of the octal numbers listed in Table 8–4.

The linking program enters the first phase of the linking process. DXCL requests that you enable the input device and, after you have, performs a partial read of the tapes to ascertain the final structure of the RTE.

The linking program enters the second phase of the linking process. It reads the input files in their entirety. It extracts the monitor and object modules from the input device as defined by the C-table, produces an executable RTE program, and writes it to the output device.

Once the linking process has successfully completed, DXCL prints/displays a message to that effect, returns to command mode, and prints/displays an asterisk (*) to prompt you for further commands.

The following is an example of using the LINK command to produce an executable RTE and write it to a directory device. The object modules are selected from an RK11 (disk drive 0), linked as defined by the C-table, and output to the same drive.

Example 8–4: Linking a Directory Device

```
*LINK DKO:TEST1.BIN<DK0:XMON??.LIB  Return
;Link Command entry
*SYS SIZE:  160000  Return
;RTE memory requirement
*MAKE OUTPUT READY. WRITE ENABLE          ;Enable output device
*TYPE(CR) WHEN READY  Return              ;Acknowledge enable
*PASS 1                                   ;Scan for all modules
*TRANSFER ADDRESS:  002200                ;RTE start address
*LOW LIMIT:  000000                       ;RTE base address
*HIGH LIMIT:  063514                      ;RTE end address
*PASS 2                                   ;Output RTE module
*LINK DONE                                ;Link process completed
```

8.2.5 General Utility Directives

You use the following directives while you are running the configurator/linker, either in or out of configure mode, to terminate a program, modify an RTE location, exit from the configurator/linker, and reboot XXDP.

8.2.5.1 Typing Control/C

You type Ctrl/C to terminate the current operation. DXCL prints/displays an asterisk (*) to prompt you for further commands. The side effects of Ctrl/C depend on the operation you have terminated.

8.2.5.2 Modifying Configurator/Linker Locations: MOD Command

You use the MOD command to modify a configurator/linker location.

The format of the command is:

MOD addr

where:

- **addr** is the physical address (octal) of the location whose contents you want to change.

DXCL responds by printing/displaying the address and its contents. You can accept or change them.

- To accept the contents, press Return.

- To change the contents, type the new value (octal) and press Return.

If you wish to examine or modify the next consecutive location (<ADDR>+2), type Line Feed.

The following is an example of using the MOD command:

```
❶   *MOD 4000 Return
❷   004000/123456 234567 Return
```

❶ Open and display location 4000

❷ The contents are 123456. Change them to 234567.

8.2.5.3 Modifying Module Headers: MOD Command

You use the MOD command to modify the contents of selected, specifically labeled words that are contained in the module interfaces (headers). You do this to make common changes in test criteria—such as, device and vector address changes and bus priority level changes.

The format of the command is:

MOD word

where:

- **word** is the octal number of the word whose contents you want to change.

Section 8.2.5.3.1 through Section 8.2.5.3.6 discuss the header words you can change and the changes you can make.

8.2.5.3.1 Word 6 (ADDR): Device/Option UNIBUS Address

Module header word 6 (ADDR) must specify the UNIBUS address of the first device or option to be tested. If more than one address is required, ADDR specifies the first address of a contiguous grouping.

The following is an example of changing header word 6 (ADDR):

```
❶ CMD> MOD WXYZ0 6 Return
❷ 52346/000000  172460 Return
```

❶ Open word 6 of module WXYZ0

❷ The location contains zeroes. Deposit the UNIBUS address 172460.

8.2.5.3.2 Word 10 (VECTOR): Device/Option Vector Address

Module header word 10 (VECTOR) must specify the vector address of the first device or option to be tested. If more than one address is required, VECTOR specifies the first address of a contiguous grouping.

The following is an example of changing header word 10 (VECTOR):

```
❶ CMD>MOD WXYZ0 10[Return]
❷ 52350/000000  230[Return]
```

❶ Open word 10 of module WXYZ0

❷ The location contains zeroes. Deposit the first vector address.

8.2.5.3.3 Word 12 (BR1,BR2): Bus Priority Levels

Module header word 12 (BR1,BR2) specifies the priority levels required by interrupt-driven devices. Normally, only BR1 is required; however, you must specify BR2 if the device has separate interrupt levels.

The following is an example of changing header word 12:

```
❶ CMD>MOD WXYZ0 12[Return]
❷ 52352/000000 300[Return]
❸  CMD>
```

❶ Open word 12 of module WXYZ0

❷ The location contains zeroes. Deposit the first BR level (PRTY6).

❸ The second BR level is unused. DXCL prompts you for further commands.

8.2.5.3.4 Word 14 (DVID1): Device Indicator Count

The number of bits that are set in module header word 14 (DVID1) indicates the total number of active devices (16 maximum) to be tested. The corresponding bit positions also specify the device(s) selected (0–15).

The following is an example of changing header word 14 (DVID1):

```
❶ CMD>MOD WXYZ0 14[Return]
❷ 52354/000000  3[Return]
```

❶ Open word 14 of module WXYZ0

❷ The location contains zeroes. Deposit 3. Device 0 (0 000 000 000) and Device 1 (000 011) are to be tested.

8.2.5.3.5 Words 16–24 (SR1–SR4): Module Switch Registers

Header words 16 through 24 (SR1, SR2, SR3, SR4) are the four 16-bit software switch registers available to each module. These registers are provided for general-purpose program switching and are used to define unique device options and/or to point to specific module routines. You can use the SRn command to deposit their contents. (See Section 8.2.3.12.)

8.2.5.3.6 Word 36 (ICONT): Iteration Constant

Module header word 36 (ICONT) indicates the number of times a module runs before an end-of-pass. The value of this word is set at the user's discretion.

The following is an example of changing header word 36:

```
❶ CMD> MOD WXYZ0 36 Return
❷ 52376/004000 100 Return
```

❶ Open word 36 of module WXYZ0

❷ The location contains zeroes. Deposit the decimal number of passes.

8.2.5.4 Exiting from the Configurator/Linker: EXIT Command

You use the EXIT command to leave the configurator/linker program and return to the XXDP monitor.

The format of the command is:

EXIT

The command does not clear the configurator/linker program from memory and does not reload the XXDP monitor. The BOOT command does. (See Section 8.2.5.5.)

8.2.5.5 Rebooting XXDP: BOOT Command

You use the BOOT command to reload the XXDP monitor.

The format of the command is:

BOOT dev:

where:

- **dev:** is the name of the device on which the XXDP monitor resides.

The following is an example of using the BOOT command:

```
*BOOT MU0: Return
```

In this example, the user loads the monitor from tape drive 0.

8.2.6 I/O Commands

You use DXCL I/O commands while you are configuring or linking the RTE to direct the listing, storage, and retrieval of an RTE's C-table and load map.

Table 8–5 lists DXCL I/O commands.

Table 8–5: I/O Commands

Command	Function
CHECK	Check object module format and checksum
GETC	Get C-table from device
PRINTC	Output C-table on line printer
PRINTM	Retrieve load map and output on line printer
SAVC	Store C-table on device
SAVM	Store load map on device
TYPEC	Output C-table on console
TYPEM	Retrieve load map and output on console

8.2.6.1 CHECK Command

You use the CHECK command to to examine an object module for proper formatting, for a checksum error, or for both. If an error is found, CHECK displays a message to that effect.

The format of the command is:

CHECK [devi:][filnam.ext]

where:

- **devi:** is the optional name of the device on which the module to be checked resides. The device can be a directory device, such as a disk, or a nondirectory device, such as a magnetic tape. If you omit this parameter, DXCL reads from the system device.
- **filnam.ext** are the optional name and extension of the object module you wish to check. This parameter is required only if the module that you want to check resides on a directory device.

8.2.6.2 GETC Command

You use the GETC command to retrieve a previously stored copy of the C-table.

The format of the command is:

GETC devi:[filnam.ext]

where:

- **devi:** is the device on which the C-table copy resides.
- **filnam.ext** are the optional name and extension of the file in which the C-table has been stored using the SAVC command. (See Section 8.2.6.5.) DXCL requires this parameter if the file resides on a directory device.

You can now use configure mode commands to modify the C-table.

8.2.6.3 PRINTC Command

You use the PRINTC command to print the contents of the current C-table on the line printer.

The format of the command is:

PRINTC

8.2.6.4 PRINTM Command

You use the PRINTM command to retrieve and print the stored copy of an RTE load map generated by a LINK command.

The format of the command is:

PRINTM [devi:][filnam.ext]

where:

- **devi:** is the optional name of the device on which the load map resides. The device can be a directory device, such as a disk, or a nondirectory device, such as a magnetic tape. If you omit this parameter, DXCL searches the system device for the load map.
- **filnam.ext** are the optional name and extension of the file in which the load map is stored. DXCL requires this parameter if the file resides on a directory device.

8.2.6.5 SAVC Command

You use the SAVC command to store a copy of the current C-table on a chosen device.

The format of the command is:

SAVC [devo:][filnam.ext]

where:

- **devo:** is the optional name of the device on which you want to store the C-table. The device can be a directory device, such as a disk, or a nondirectory device, such as a magnetic tape. If you omit this parameter, DXCL stores the current C-table on the system device.
- **filnam.ext** are the optional name and extension of the file in which you want to store the C-table's contents. DXCL requires this argument only when it writes to directory devices.

You can now retrieve the C-table for modification or reuse.

8.2.6.6 SAVM Command

You use the SAVM command to store a copy of the load map generated by a LINK command on a chosen device.

The format of the command is:

SAVM [devo:][filnam.ext]

where:

- **devo:** is the optional name of the device on which the load map is to be stored. The device can be a directory device, such as a disk, or a nondirectory device, such as a magnetic tape. If you omit this parameter, DXCL stores the load map on the system device.
- **filnam.ext** are the optional name and extension of the file in which the load map is stored. DXCL requires this parameter when it stores the load map on a directory device.

You must issue this command immediately following the configurator/linker's `LINK DONE` message.

8.2.6.7 TYPEC Command

You use the TYPEC command to type the contents of the current C-table on the system console.

The format of the command is:

TYPEC

8.2.6.8 TYPEM Command

You use the TYPEM command to retrieve the stored copy of a load map generated by a LINK command and type it on the system console.

The format of the command is:

TYPEM [devi:][filnam.ext]

where:

- **devi:** is the optional name of the device on which the load map is stored. The device can be a directory device, such as a disk, or a nondirectory device, such as a magnetic tape. If you omit this parameter, DXCL searches the system device for the load map.
- **filnam.ext** are the optional name and extension of the file in which the load map's contents are stored. DXCL requires this parameter when it searches a directory device for the file.

Chapter 9
Batch Control

9.1 Overview

You can use XXDP to run diagnostics without manual intervention. This is called batch control (or chaining), and the command files that you set up to run diagnostic programs are called batch (or chain) files.

Batch files are text files into which you write commands that you would normally type at a terminal one by one. (You use XTECO to write the files.)

You run the batch file, and XXDP processes the commands it contains, so you do not have to enter each one manually.

Once you have created a batch file, you can use it over and over again. You no longer have to do repetitive tasks, such as building new media or running a commonly used set of diagnostics.

Batch control lets you develop a test strategy and use the strategy consistently. You do so by selecting the proper diagnostics and running them in a particular order and mode to achieve the best test process. Once you have worked out such a process, you can put it into a batch file.

9.1.1 Batch Control Functions

Table 9–1 lists XXDP batch control functions and the commands batch files can contain.

Table 9–1: Batch Control Functions

Command Type	Commands
DRS commands	All DRS commands and diagnostic dialogue
Monitor commands	CHAIN, ENABLE, LOAD, RUN, and START commands
Utility commands	UPDAT, SETUP, and PATCH commands

Section 9.1.2 through Section 9.1.4 briefly discuss these functions.

9.1.2 DRS Commands

You can place all DRS commands and their switches and flags in batch files. (Chapter 3 discusses these commands and switches.) You can include all dialogue that would normally take place between an operator and a DRS diagnostic in a batch file.

9.1.3 Monitor Commands

You can place certain of the monitor commands described in Chapter 2 in batch files. They are the CHAIN, ENABLE, LOAD, START, and RUN commands. The ENABLE, LOAD, and START commands function under batch control as they would under operator control; however, the CHAIN and RUN commands function differently when you put them in a batch file:

- The RUN command has a pass switch that you can specify when you are running diagnostic programs that are not DRS-compatible. The following is an example of using the pass switch:

  ```
  R DIAG/5
  ```

  DIAG runs five passes before XXDP executes the next command in the batch file. (See Appendix E for further information.)

- You can place the CHAIN command in a batch file, with one restriction: you can nest batch operations only once. A batch file can start another batch file and continue after the second file has been processed; however, the second batch file cannot start a third file.

9.1.4 Utility Commands

You can use batch files to run XXDP utilities. Batch control processes utility commands.

9.2 Special Batch Control Directives

You place the following directives in a batch file in addition to DRS, monitor, and utility commands.

Command Type	Function
Conditionals	Process sections of the batch file conditionally, depending on operator input or runtime conditions
GOTO tag	Begin processing at another section of the batch file designated by tag

Command Type	Function
PRINT	Temporarily override QUIET
QUIET	Inhibit/enable printing of batch file progress and errors
QUIT	Terminate the batch operation
SMI/CMI	Enable/disable manual intervention operations in specialized diagnostics
WAIT	Stop batch operation until the operator types Ctrl/X

Section 9.2.1 through Section 9.2.7 discuss using special batch control directives.

9.2.1 Conditionals

A batch file can make sure that XXDP does or does not process sections it contains depending on your input or on the existence of certain conditions. Batch files set up conditions using one of these directives:

-
  ```
  IF condition
  THEN statement(s)
  END
  ```

-
  ```
  IFERR THEN
  statement(s)
  END
  ```

-
  ```
  IFLMD n THEN
  statement(s)
  END
  ```

9.2.1.1 IF Condition THEN

If a batch file contains the following code,

```
IF condition
THEN statement(s)
END
```

and the condition specified is true, XXDP processes the statements between THEN and END. If the condition is false, XXDP ignores these statements.

The conditions are ASCII character strings that have been defined in the batch file. You specify these as switches when you run the batch file. For example, suppose that part of the program run under batch control requires the presence of an RX02 on the system. The batch file would contain a conditional such as the following to find out whether an RX02 is present.

```
IF RX02 THEN
statement(s)
END
```

You would add the switch RX02 to the CHAIN command as follows to indicate that an RX02 is present,

```
C FILE/RX02
```

and the batch file would process the RX02 code. If you did not add the switch, the batch file would not process it.[1]

9.2.1.2 IFERR Condition THEN

If a batch file contains the following code,

```
IFERR THEN
statement(s)
END
```

XXDP processes the statements between THEN and END—if the last DRS-type diagnostic run under batch control detected a test error. If no error was detected, XXDP ignores these statements.

9.2.1.3 IFLMD n THEN

If a batch file contains the following code,

```
IFLMD n THEN
statement(s)
END
```

XXDP processes the statements—if the media-type byte in physical location 41 contains a type code that matches the one specified in the conditional "n".

9.2.2 GOTO Label

Batch files use the GOTO directive to transfer control within a batch file to the place indicated in the command. For example, if the batch file contains the following code,

```
GOTO MMUTEST:
```

XXDP searches for the lines that begin with MMUTEST: and processes the statements(s) that follow the tag. The tag can occur before or after the GOTO statement in the batch file. The following are examples of the GOTO statement:

[1] The only predefined batch control switch is /QV (quick verify). You add it when you want the diagnostics contained in a batch file to be run for only one pass.

```
TAG1:
R PROG1
GOTO TAG1
```

In the above example, the batch process loops backwards until interrupted by the operator.

```
GOTO TAG2
    .
    .
    .
TAG2:
R PROG5
```

In the above example, XXDP transfers control forward to TAG2 and ignores statements between GOTO and the tag.

9.2.3 PRINT Command

Batch files contain the PRINT command to force the printing/displaying of a line of text while printing/displaying is inhibited by the QUIET directive. (See Section 9.2.4.)

The format of the directive is:

PRINT text

XXDP prints/displays the text that is located on the same line as the PRINT directive.

9.2.4 QUIET Directive

Batch files contain the QUIET directive to control typing of their contents. The directive acts as a "flip-flop." The first time XXDP finds the directive in a batch file, it inhibits printing/displaying (with the exception of error messages). The next time XXDP finds the directive, it reenables printing/displaying. The third time, it inhibits them, and so on.

9.2.5 QUIT Directive

Batch files contain the QUIT directive to stop the batch job and return to normal monitor mode.

9.2.6 SMI/CMI Directives

Batch files contain the SMI and CMI directives to enable and disable manual (operator) intervention modes in DRS-type diagnostics.

Normally, testing that requires operator intervention is inhibited during batch control operations. The SMI and CMI directives allow batch files to override this inhibition.

SMI means allow manual intervention. CMI means do not allow manual intervention. CMI is the default state when a batch job is started.

9.2.7 WAIT Directive

Batch files contain the WAIT directive to stop XXDP from processing their contents and to wait for you to perform manual functions (such as placing a scratch disk in a drive). Once you have finished, you type Ctrl/X and XXDP starts the batch file again.

The following is an example of the batch control feature that stops and waits:

```
PRINT THE NEXT DIAGNOSTIC REQUIRES THAT A
PRINT SCRATCH MEDIUM BE MOUNTED IN THE RL01/02.
PRINT TYPE ^X WHEN READY
WAIT
R ZRLA??
```

9.3 Comments

Batch files can contain comments. XXDP prints/displays these as it processes the file, unless QUIET mode has been invoked.

A comment is a string of text that starts with a semicolon (;). The following is an example of a comment contained in a batch file:

```
;THE NEXT PROGRAM TESTS THE DZ11
;
R ZDZB??       ;RUN THE DIAGNOSTIC
```

9.4 Batch Control of Diagnostics

Batch control can process two types of diagnostics:

1. Chainable non-DRS-type
2. DRS-type

You run them as follows:

1. Issue a RUN command to run the first type, as follows:

   R DIAG[/n]

   where:

   - **n** is an optional argument that specifies the number of passes that the diagnostic will run. If you omit this number, the diagnostic runs for one pass.

2. DRS-type diagnostics require complete batch control. The batch file must contain all commands that the operator normally issues and reply to all prompts that the diagnostic normally issues. You run this batch file. The following is an example of batch control lines containing answers to prompts:

   ```
   R DIAG2
   START/PASS:1
   Y       [answer for CHANGE HW]
   1       [answer for number of units]
   [insert answers for all HW questions]
   EXIT    [to return control to batch job]
   ```

   If the diagnostic program in the above example had used a software table, it would have been necessary to provide the commands required to support it.

You can use the SETUP utility before you run a batch job to supply necessary hardware and software information to the diagnostic. Appendix E discusses prebuilding the tables that supply this information for use in batch files. Digital recommends this method for running DRS-type diagnostics in the batch control environment. If you preset all the parameters, you place only the following commands in your batch file:

```
R DIAG2
START/PASS:n
N
N
EXIT
```

where:

- **n** is the number of passes to execute.

9.4.1 Diagnostic Halt and Stop

You can temporarily halt a DRS diagnostic contained in a batch file by typing Ctrl/C at any point in its flow.

The following code tests for Ctrl/C and reacts accordingly:

```
R DIAG
N
N
EXIT
IF _^C THEN
GOTO L1
END
```

You can permanently stop a batch file by typing Ctrl/Z.

9.4.2 Batch Control of Utilities

You can run UPDAT, SETUP, and PATCH under batch control. Create a batch file that contains all the utility commands that an operator normally issues.

The following is an example of a batch file that builds an RX01 floppy disk for use by XXDP:

```
R UPDAT
ZERO DU0:
CREATE DU0:
FILE DU0:=DRSXM.SYS
FILE DU0:=DIR.SYS
FILE DU0:=DY.SYS
EXIT
```

The dialogue with UPDAT must end with an EXIT command, so that the batch job can be finished or further batch functions can be carried out.

Appendix A Command Summary

This appendix summarizes the following:

- Monitor commands
- DRS commands, switches, and flags
- UPDAT commands
- PATCH commands
- SETUP commands
- XTECO commands
- DXCL commands
- Batch control functions

A.1 Monitor Commands

The following tables summarize the SM and XM monitor commands.

Table A–1: SM Monitor Commands Summary

Command	Function
Chain	Run a batch job (chain)
Directory	List directory of load media
Enable	Enable alternate system device
Help	Type help information
Load	Load a program
Run	Run a program
Start	Start a program
VT	Change terminal type

Table A–2: XM Monitor Commands Summary

Command	Function
Boot	Boot a device
BOOTSM	Boot small monitor
Chain	Run a batch job (chain)
CLEAR	Return monitor to a neutral state (negates DEFSM and DEFXM)
COpy	Copy a file or device
DAte	Set or show the date
DEFSM	Set default operation to small monitor
DEFXM	Set default operation to extended monitor
DElete	Delete a file
Directory	List directory of load media
Enable	Enable alternative drive as system device
Help	Print/display help information
Initialize	Initialize a device
Load	Load a program
Print	Print a file on system line printer
REname	Rename a file to a new name
Run	Run a program
SEt	Set device or system parameter
Start	Start a program
TEST	Enter user friendly mode
Type	Type a file
VERSION	Print information about extended monitor

A.2 DRS Commands

The following table summarizes the DRS commands.

Table A–3: DRS Commands Summary

Command	Function
ADD	Activate a unit for testing
CONtinue	Continue diagnostic at test that was interrupted by a Ctrl/C
DISplay	Print a list of device information
DROp	Deactivate a unit
EXIt	Return to XXDP runtime monitor
FLAgs	Print status of all flags
PROceed	Continue from an error halt
REDirect	Redirect error prints and statistics to another unit and/or line printer[1]
REStart	Start diagnostic and do not initialize
STArt	Start diagnostic and initialize
ZFLags	Reset all flags

[1]DRSXM only.

A.3 DRS Command Switches

The following table summarizes the DRS command switches.

Table A–4: DRS Switches Summary

Switch	Function
/EOP:ddddd	Report end-of-pass after each ddddd passes (ddddd = 1 to 64000)
/FLAGS:flag-list	Set specified flags
/PASS:ddddd	Execute ddddd passes (ddddd = 1 to 64000)
/TESTS:test-list	Execute only tests specified
/UNITS:unit-list	Command will affect only specified units

A.4 DRS Flags

The following table summarizes the DRS flags.

Table A–5: DRS Flags Summary

Flag	Effect
ADR	Execute autodrop code
BOE	Bell on error
EVL	Execute evaluation
HOE	Halt on error
IBE	Inhibit all error reports except first level
IDU	Inhibit program dropping of units
IER	Inhibit all error reports
ISR	Inhibit statistical reports
IXE	Inhibit extended error reports
LOE	Loop on error
LOT	Loop on test
PNT	Print test number as test executes
PRI	Direct messages to line printer
UAM	Unattended mode (no manual intervention)

A.5 UPDAT Commands

The following table summarizes the UPDAT commands.

Table A–6: UPDAT Commands Summary

Command	Function
ASG	Assign a logical name to a device
BOOT	Bootstrap a device
CLR	Clear UPD2 program buffer
COPY	Copy entire media
CORE	Print limits of buffer addresses

Table A–6 (Cont.): UPDAT Commands Summary

Command	Function
CREATE	Place a bootable monitor on a media
DEL	Delete a file(s)
DIR	Give directory of specified media
DO	Execute an indirect command file
DRIVER	Load a device driver
DUMP	Dump a program image
EOT	Write logical end-of-tape mark on a tape
EXIT	Return control to runtime monitor
FILE	Transfer a file(s) and autodelete
HICORE	Set upper memory limit for dump
LOAD	Load a program
LOCORE	Set lower memory limit for dump
MOD	Modify file image in memory
PIP	Transfer a file(s) without autodeletion
PRINT	Print a file on line printer
READ	Read a file to check validity
REN	Rename a file
TYPE	Type a file on the console terminal
XFR	Set transfer address
ZERO	Initialize a media

A.6 PATCH Commands

The following table summarizes the PATCH commands.

Table A–7: PATCH Commands Summary

Command	Function
BOOT	Boot specified device
CLEAR	Clear input table
EXIT	Return to XXDP monitor
GETM	Load DEC/X11 MAP file
GETP	Load saved input table
KILL	Delete address from input table
MOD	Enter address in input table
PATCH	Create patched file
SAVP	Save input table
TYPE	Print input table on terminal

A.7 SETUP Commands

The following table summarizes the SETUP commands.

Table A–8: SETUP Commands

Command	Function
EXIT	Return control to XXDP
LIST	Print/display a list of DRS diagnostics on a media
SETUP	Build tables for specified diagnostic

A.8 XTECO Commands

The following table summarizes the XTECO commands.

Table A–9: XTECO Commands Summary

Command	Function
A	Append text to that currently in memory
C	Move cursor character by character
D	Delete character(s)
EDIT	Modify a file
EXIT	Return to monitor
EX	Exit edit mode
I	Insert text
J	Move cursor to beginning of text
K	Delete line(s)
L	Move cursor line by line
N	Search for specified string in remainder of text file
PRINT	Print a file on line printer
S	Search for specified string in text now in memory
T	Type text
TECO	Modify a file on disk
TEXT	Create new text file
TYPE	Type a file on console terminal
ZJ	Move cursor to end of text

A.9 DXCL Commands

The following table summarizes DXCL commands. The asterisk (*) denotes commands valid only in command mode.

Table A–10: DXCL Keyboard Commands

Command	Function
COFF	Disable cache memory
CON	Enable cache memory
DES	Deselect all modules
DES modulename	Deselect named module
EXAM	Output last examined location
EXAM addr	Output specified location for examination
EXAM modulename addr	Output specified location in named module for examination
FILL	Output contents of FILL CHAR/FILL CNT location
FILL number number	Replace contents of FC/FC location and output same
*KTOFF	Disable memory management
*KTON	Enable memory management
LPOFF	Disable console output to line printer
LPON	Enable console output to line printer
MAP	Output maps for all modules
MAP modulename	Output map for named module
*MOD	Output contents of last modified location
*MOD addr	Output contents of address specified
*MOD modulename addr	Output contents of address specified in named module
*MOFF	Disable map box
*MON	Enable map box
PON	Enable parity memory
POFF	Disable parity memory
ROTOFF	Disable write buffer rotation
ROTON	Enable write buffer rotation

Table A–10 (Cont.): DXCL Keyboard Commands

Command	Function
*RUN	Execute exerciser
*RUN addr	Execute exerciser at specified address
*RUNL	Lock and execute exerciser
*RUNL addr	Relocate to specified address, lock, and execute exerciser
SEL	Select all modules
SEL modulename	Select named module
SUM	Output summary message for all modules
SUM modulename	Output summary message for named module
SWR	Output contents of software switch register (SR)
SWn number	Replace contents of SR and output same

A.10 Batch Control Functions

The following table summarizes XXDP batch control functions.

Table A–11: Supported Batch Control Functions

Command Type	Supported Commands
Monitor commands	CHAIN, ENABLE, LOAD, RUN, and START commands
Utility commands	UPDAT, PATCH, and SETUP commands
DRS commands	All DRS commands and diagnostic dialogue
Conditionals	Sections of batch file processed, depending on operator input or runtime conditions (e.g., IF THEN)
GOTO tag	Processing beginning at section of batch file designated by tag
QUIET	Printing of batch file inhibited/enabled if printing was previously enabled/inhibited
PRINT	Temporary override of QUIET
SMI/CMI	Manual intervention enabled/disabled in specialized diagnostics
QUIT	Termination of batch operation
WAIT	Suspension of batch operation until operator types a Ctrl/X

Appendix B
Supported Devices

XXDP supports most mass storage devices. The following table lists each supported device, its mnemonic, corresponding driver, and distribution media (when relevant).

Table B–1: Devices Supported

Device	Mnemonic	Driver	Distribution Media
PRINTER	LP	LP	
RD/RX	DU	DU	RX50
RK06/7	DM	DM	RK06/RK07
RL01/2	DL	DL	RL02
RM02/3	DR	DR	
RP04/5/6	DB	DB	
RX02	DY	DY	RX02
TK50/TU81	MU	MU	TK50
TM02/3	MM	MM	800 BPI MT
TSV05/TU80	MS	MS	1600 BPI MT
TU58	DD	DD	TU58
UDA50	DU	DU	

All drivers assume that the CSR address for the device is the standard address, as given in the peripheral handbook for your processor. If you have a system with a device at a nonstandard address, use UPDAT to modify location 12 in the driver.

XXDP device drivers are small and limited in function. They can detect and report three types of errors: read, write, and hard (unrecoverable). After drivers report these errors, they return control to the utility that called them. The utility takes required action. Driver functionality is limited: if the error persists, the user has to run appropriate diagnostics.

Appendix C
Component Names

Table C–1 lists XXDP file names.

Table C–1: XXDP File and Release Names

Description	File Name
MONITOR Files	
XXDP V2 DATE UTILITY	DATE.SYS
XXDP V2 DIRECTORY UTILITY	DIR.SYS
XXDP V2 EXTENDED MONITOR	XXDPXM.SYS
XXDP V2 HELP FILE	HELP.TXT
XXDP V2 RESIDENT MONITOR	XXDPSM.SYS
DRIVER Files	
MSCP DRIVER & BOOT	DU.SYS
PRINTER DRIVER	LP.SYS
RK06/7 DRIVER & BOOT	DM.SYS
RL01/2 DRIVER & BOOT	DL.SYS
RM02/3 DRIVER & BOOT	DR.SYS
RP04/5/6 DRIVER & BOOT	DB.SYS
RX02 DRIVER & BOOT	DY.SYS
TK50 DRIVER & BOOT	MU.SYS
TM02 DRIVER & BOOT	MM.SYS
TS04/TS11 DRIVER & BOOT	MS.SYS
TU58 DRIVER & BOOT	DD.SYS

Table C–1 (Cont.): XXDP File and Release Names

Description	File Name
UTILITY Files	
XXDP V2 DECX11 CONFIGURATOR AND LINKER	DXCL.BIC
XXDP V2 PATCH UTILITY	PATCH.BIC
XXDP V2 SETUP UTILITY	SETUP.BIC
XXDP V2 UPDATE UTILITY	UPDAT.BIC
XXDP V2 XTECO UTILITY	XTECO.BIC
UFD Files	
CLEARS LINE CLOCK FOR TK25 USE	HCLR??.BIC
DATA BASE TEXT FILE	HDBA??.UFD
ERROR STACK LOADER, LOC.44 TO STK	HUXE??.BIC
EXT. OPT. LANG FILE FOR HUCN??.BIN	HCNA??.UFD
FILE THAT DOES RESET INSTRUCTION	HRST??.BIC
INDIVIDUAL DEVICE TEST BUILDER	HUSE??.BIC
LANGUAGE FILE FOR HUSE??.BIC	HSEA??.UFD
SYSTEM SIZER MODULE	HSIZ??.BIC
UFD CONFIGURATION MOD	HUCN??.BIC
UFD LINKER MOD	HULN??.BIC
XXDP STACK HANDLER MODULE	HSTK??.BIC
XXDP V2 UFD MENU UTILITY	MENU?0.BIC
DRS Files	
XXDP DIAG SUPR EXT	DRSXM.SYS
XXDP DIAG SUPR SML	DRSSM.SYS

Appendix D
Building XXDP

D.1 Building a Bootable System

The minimum files that must be put on a bootable XXDP media are as follows:

- Monitor corresponding to that media
- Device driver for that media
- DRS (DRSSM.SYS or DRSXM.SYS)
- Directory utility (DIR.SYS)

To create a bootable XXDP media:

1. Run UPDAT
2. Use the UPDAT CREATE command to place the monitor file on the new media
3. Use the appropriate UPDAT file transfer commands to place the remaining files on the media

Chapter 4 discusses UPDAT and its commands in detail.

The following are examples of building the minimum XXDP system on an RX02 and a TS04, respectively:

```
.R UPDAT

❶ CREATE DY0:

❷ FILE DY0:=DY.SYS

❸ FILE DY0:=DIR.SYS

❹ FILE DY0:=DRSXM.SYS

EXIT
```

```
.R UPDAT
❶ CREATE MS0:
❷ PIP MS0:=MS.SYS
❸ PIP MS0:=DIR.SYS
❹ PIP MS0:=DRSXM.SYS
EXIT
```

❶ Place a bootable monitor on devices DY0: and MS0:, respectively

❷ Place the correct driver and boot on the device

❸ Place the directory utility on the device

❹ Place the DRS extended monitor on the device

You can add as many other system components (drivers and utilities) as you wish.

D.1.1 Creating a New XXDP Media (Using UPDAT)

NOTE: *For DD, substitute device name; For ??, substitute revision and patch level.*

```
❶ .INIT DD1:
❷ .CREATE DD1:
❸ .LOAD DD0:DD.SYS
❹ .DUMP DD1:DD.SYS
❺ .PIP DD1:DIR.SYS DD0:DIR.SYS
❻ .PIP DD1:HSAA??.SYS= DD0:HSAA??.SYS
❼ .PIP DD1:UPDAT.BIN= DD0:UPDAT.BIN
❽ .PIP DD1:XTECO.BIN= DD0.XTECO.BIN
```

❶ Zero directory on new media

❷ Save in monitor area of new media

❸ Load device driver

❹ Save in program area of new media

❺ Transfer directory program

❻ Transfer diagnostic supervisor

❼ Transfer UPDAT utility

❽ Transfer XTECO utility

Use the PIP command to copy as many other monitor and driver files as desired to the new media.

Appendix E User Tips

Appendix E contains some tips you should keep in mind when running XXDP.

E.1 Table Building

Prebuilding hardware and software tables saves you time and energy.

Take advantage of the SETUP utility to prebuild the hardware and software tables you will use in a diagnostic. Customize the files for a specific system on the corresponding XXDP media or customize files for several systems on media shared between systems.

You can use the START command to change the tables at run time as needed. You can use SET to change the files permanently.

E.2 Using Batch Control

Using batch control saves you time and energy.

Familiarize yourself with the DRS-type diagnostics for a particular device and identify the various operating modes (as defined by the software tables) that are most useful for you. Prebuild the hardware tables for the system(s) you are working with. Then write a batch control file that implements the various operating modes based on conditionals. You can now enter one command to XXDP and then let the system do the rest.

The following is an example of running a diagnostic file interactively. The file builds a fictional diagnostic called DIAG1. If you START this diagnostic, which already has hardware tables, the dialogue is as follows:

```
   .R DIAG1
   DR> STA
   CHANGE HW (L) ?  N
   CHANGE SW (L) ?  Y
❶  TEST ALL SECTORS (L) ?
```

❶ The diagnostic can test some or all sectors of the UUT and prompts you for the option you want. Your answer becomes part of the software table.

You can create a batch file to run DIAG1 and answer the questions. You can call the file DISK.CCC. The file defines two test modes, QV (quick verify) and REPAIR, to answer the software question. If you choose QV, the diagnostic tests some sectors. If you choose REPAIR, the diagnostic tests all sectors.

```
❶  IF QV THEN
    R DIAG1
    STA/PAS:1
    N
    Y
    N
    END
❷  IF REPAIR THEN
    R DIAG1
    STA/FLA:LOE
    N
    Y
    Y
    N
    END
```

❶ If the test mode is QV, DISK.CCC runs the tests contained in DIAG1 for one pass.

❷ If the test mode is REPAIR, DISK.CCC runs the tests contained in DIAG1 indefinitely and directs it to loop on error.

When you run the batch file, you add a switch specifying either QV or REPAIR mode, and the batch file executes code accordingly.

The following are examples of running DISK.CCC in QV and REPAIR mode:

```
❶  C DISK/QV
❷  C DISK/REPAIR
```

E.3 Running Diagnostics from Tape Media

You may encounter problems when you run certain diagnostics from a tape media. Some diagnostics may cause the system to respond very slowly and others may cause the system to fail.

If you think that you have encountered one of these diagnostics, use the BOOTSM command to boot the small monitor and rerun the diagnostic.

E.4 RTE Debugging Recommendations

Here is a checklist to help you analyze and isolate faults that may occur in a newly created runtime exerciser (RTE) program.

If errors occur during the testing of a newly created RTE program:

- Check software parameters—such as VCT, BR1, SR1, and so on
- Eliminate the possibility of a peripheral error by changing tapes, cleaning heads, changing disk packs, and so on
- Run a stand-alone diagnostic if a device failure is indicated
- Run the RTE on another system (if practical) if hardware failures are persistent and varied
- Use the RUNL command to the program locked in different banks if multiple module failures occur
- Run a module alone or in different combinations with others if it fails

Appendix F
RTE Device/Option Modules, Software Switch Options, and Sample Build

F.1 RTE Device/Option Modules

The following table summarizes available RTE device/option modules.

Table F–1: Device Option List

Device/Option	Test Module
AA11-K	AAB
AA11/VT01-A	AAA
AAV-11	AAC
AD01-D	ADA
AD11-K	ADB
ADV-11	ADC
AFC11	AFA
AR-11	ARA
BDV11/KDF11-B	BMD
BM792-YA	BMC
BM873-SC, SE, SF	BMC
BM873-YA, YB, YC, YD	BMC
BM873-YF	BME
BM873-YH	BMF
BM873-YJ	BMG
BUS EXERCISER (Q-BUS)	BTC

Table F–1 (Cont.): Device Option List

Device/Option	Test Module
BUS EXERCISER (UNIBUS)	BTA, BTB
CB-11	CBC
CB-11 (SCAN)	CBA
CB-11 (DISTRIBUTE)	CBB
CD-11	CDA
C.I.S (11/24, 11/44)	CIA
CM-11	CRA
CMS-11K	CMA
CMR-11	CMJ
CR-11	CRA
DC-11	DCA
DH-11	DHA
DJ-11	DJA
DL-11	DLA
DL-11E	DLB
DM11-BA	DMS
DM11-BB	DMB
DMC-11	DMC
DMP-11 (MASTER)	DMD
DMP-11 (SLAVE)	DME
DMR-11	DMR
DN11	DNA
DP11	DPA
DPV11	DPV
DQ11	DQA
DR11-A	DRA
DR11-B	DRB

Table F–1 (Cont.): Device Option List

Device/Option	Test Module
DR11-C, DR11-K	DRC
DR11-K	DRD
DR11-L, DR11-M	DRE
DR11-W	DRW
DR70	DRK
DRV11-B	DRF
DRV11-J	DRJ
DTE20	DTA
DU11	DUA
DUP11	DPB
DV11	DVA
DX11	DXA
DZ11	DZA
DZV11	DZB
E.I.S	CPB
FIS(40, LSI)	FPA
FP11-A, B, C, 11/60	FPB
FP11-C(40/45)	FPA
GROSS TIMING	KWF
GT40	GTA
IBV11-A	IBA
ICR-11	ICB
ICS-11	ICA
INSTRUCTIONS	CPA
KE11	KEA
KG11	KGA
KIT11D	BBA

Table F-1 (Cont.): Device Option List

Device/Option	Test Module
KL11	KLA
KMC11	KMC
KMC11-B	KMA
KUV11-AA	KUA
KW11-C	KWG
KW11-K	KWD
KW11-L	KWA
KW11-P	KWB
KW11-W	KWC
KWV11-K	KWE
LK-11	LKA
LP11	LPA
LP20/LP05/LP10	LPF
LPA11	LPH
LPD	LPE
LPS11/LPS-AD/NP	LPD
LPS11/LPS-KW	LPB
LPS11/LPS-VC	LPC
M7855 BUS TESTER	BEA
M7942-YB	BMI
M9301-YA, B, C, D, E, F, H, J	BMI
M9301-SA, SB	BMC
M9311	BMI
M9312	BMH
M9400	BMC
YA, YC, YH, YK, YN	BMI
ML11-A	MLA

Table F-1 (Cont.): Device Option List

Device/Option	Test Module
MNCAD	MNA
MNCDA	MND
MNCDI	MDB
MNCDO	MNE
MNCKW	MNC
MR11	BMC
NC11A	NCA
NCV-11A	NCB
PA611 READER	PAA
PA611 PUNCH	PAB
PC11	PCC
PCL-11	PLA
PCS-11	PCS
RA80/UDA50	DUB
RC11/RS64	RCA
RF11	RFA
RH01	RHA
RH11/RH70 SGL PT DSK	RPB
RK11/RK02, 03, 04, 05	RKA
RK611/RK06, 07	RKB
RKV11	RKA
RL11/RL01	RLA
RM02/O3 RH11/70	RMA
RM03/RH11	RMB
RM02, 3, 5 RH11/70	RMC
RM02, 03, 05	RMD
RM80	RNA

Table F–1 (Cont.): Device Option List

Device/Option	Test Module
RP04, 05, 06	RMD
RP11	RPA
RS03, RS04/RH11	RSA
RX01	RXA
RX02	RXB
TA11	TAA
TC11	TCA
TM02, TM03, TE16, TU16, TU16-EK, TU77	TMB
TM11	TMA
TM78	TMD
TR79F	TRA
TS04	TSA
TU58	TUA
UDA-50/RA80	DUB
UDC11	UDA
VS11, VSV11	VSC
VS60	VSA
VSV01	VSB
VT20	VTA
VT20 on DH11	VTB
VTV30-K	VTC
VT30-H	VTV
VTV30-J/H	VTV
XY11	XYA
YA, YB, YC, YF, YH	BMC

F.2 Switch Register Options

The DXCL monitor provides a software switch register for system use. Thus, do not use hardware switch registers.

Table F–2 lists the SR bits that you can set or clear to provide these features.

Table F–2: Software Switch Register Bits

Bit	Operation
SR00 = 0	Disable printing of 1-character NULL message
SR00 = 1	Enable printing of 1-character NULL message
SR08 = 0	Cycle exerciser once through all of memory; then allow random relocation
SR08 = 1	Cycle exerciser through memory by constant offset value, while inhibiting random relocation
SR09 = 0	Enable RELOCATED TO printout
SR09 = 1	Inhibit RELOCATED TO printout
SR10 = 0	Report only first 3 data errors occurring within a transferred block
SR11 = 1	Report all data errors
SR12 = 1	Permit end-of-pass printouts
SR13 = 1	Inhibit error and module printouts
SR14 = 0	After 20th error, and following a MODULE DROPPED printout, drop module
SR14 = 1	After 20th error, inhibit dropping of module
SR15 = 1	After 1 error (hard or soft), and following a MODULE DROPPED printout, drop module

F.3 Sample RTE Build

The following shows you the steps in a sample RTE build.

F.3.1 Prebuild Planning

The system configuration for building a runtime exerciser (RTE) is as follows:

- PDP-11/70 processor
- 256K of memory
- Extended instruction set

- Cache
- Floating point hardware
- 1-RM03 single port disk
- M9312
- 2-TM03/TE16
- 1-LP11
- 1-RS04
- 1-DH11

The corresponding configuration worksheet follows:

Figure F–1: DEC/X11 System Configuration Worksheet

```
Selected DEC/X11 Monitor for Listed
CPU and CPU options:E

            FILE:  ESAMC0.BIN    DATE:  20 SEPT 87
```

DEVICE	MOD	R	DVA	VCT	BR1	BR2	DVC	SR1	SR2	SR3	SR4
RM03	RMD	A	176700*	254*	5*	0*	1*				
LP11	LPA	A	177514*	200*	4*	0*	1*	77000			
TM03/TE16	TMB	A	172440*	224*	5*	0*	2				
RS04	RSA	A	172040*	204*	5*	0*	1*				
DH11	DHA	A	160200	300	5*	5*	1*				
EIS	CPB	A									
11/34 Instr.	CPA	A									
FP11-C	FPB	A									
M9312	BMH	A									

```
*   DENOTES SOFTWARE DEFAULT PARAMETERS
```

F.3.2 Configuring the RTE

You now build the configuration table (C-table).

Boot XXDP, run DXCL, and run the configurator/linker.

```
$DK0 [Return]                               ;Boot the Load Media
CHMDKB0 XXDPXM DK MONITOR
BOOTED VIA UNIT#:  0
28K UNIBUS MEMORY
ENTER DATE (DD-MMM-YY):
RESTART ADDR:152010
THIS IS XXDP  TYPE "H" OR "H/L" FOR HELP.
TYPE:  <^C>                                 ;Abort XXDP+ Header Message
.R DXCL                                     ;Run the configurator/linker program

CHUXCC0 XXDP DEC/X11 CNF/LNK
RESTART:  006472
DO YOU WANT HELP?(Y <CR> OR JUST <CR>)  [Return] ;Inhibit help message
```

F.3.2.1 Building the C-Table

In this example, the DXCL prompts you for module entry information.

```
*CNF [Return]                               ;Enter CNF mode
MONITOR:  E [Return]                        ;Enter Monitor name
*MDL RMDA [Return]                          ;Enter Module RMDA
DVA- [Return]
VCT- [Return]
BR1- [Return]
BR2- [Return]
DVC- [Return]
SR1- [Return]
SR2- [Return]
SR3- [Return]
SR4- [Return]
RMAA  DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
 SR1-000000 SR2-000000 SR3-000000 SR4-000000
```

```
*MDL LPAA [Return]                              ;Enter Module LPAA
DVA- [Return]
VCT- [Return]
BR1- [Return]
BR2- [Return]
DVC- [Return]
SR1-77000 [Return]                              ;Change LPAA SR1 value
SR2- [Return]
SR3- [Return]
SR4- [Return]
LPAA  DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
 SR1-077000 SR2-000000 SR3-000000 SR4-000000

*MDL TMBA [Return]                              ;Enter Module TMBA
DVA- [Return]
VCT- [Return]
BR1- [Return]
BR2- [Return]
DVC-2 [Return]                                  ;Change TMBA DVC value
SR1-40 [Return]                                 ;Change TMBA SR1 value
SR2- [Return]
SR3- [Return]
SR4- [Return]
TMBA  DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000003
  SR1-000040 SR2-000000 SR3-000000 SR4-000000
*MDL RSAA [Return]                              ;Enter Module RSAA
DVA- [Return]
VCT- [Return]
BR1- [Return]
BR2- [Return]
DVC- [Return]
SR1- [Return]
SR2- [Return]
```

```
SR3- [Return]

SR4- [Return]

RSAA  DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
 SR1-000000 SR2-000000 SR3-000000 SR4-000000

*MDL DHAA [Return]                 ;Enter Module DHAA

DVA-160200 [Return]                ;Change DHAA DVA value

VCT-300 [Return]                   ;Change DHAA VCT value

BR1- [Return]

BR2- [Return]

DVC- [Return]

SR1- [Return]

SR2- [Return]

SR3- [Return]

SR4- [Return]

DHAA  DVA-160200 VCT-000300 BR1-000000 BR2-000000 DVC-000000
 SR1-000000 SR2-000000 SR3-000000 SR4-000000

*MDL CPBA [Return]                  ;Enter Module CPBA

DVA- [Return]

VCT- [Return]

BR1- [Return]

BR2- [Return]

DVC- [Return]

SR1- [Return]

SR2- [Return]

SR3- [Return]

SR4- [Return]

CPBA  DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
 SR1-000000 SR2-000000 SR3-000000 SR4-000000

*MDL CPAA [Return]                 ;Enter Module CPAA

DVA- [Return]

VCT- [Return]
```

```
BR1- [Return]

BR2- [Return]

DVC- [Return]

SR1-77000 [Return]

SR2- [Return]

SR3- [Return]

SR4- [Return]

CPAA  DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
 SR1-077000 SR2-000000 SR3-000000 SR4-000000

*MDL FPBD [Return]                      ;Enter Module FPBD

DVA- [Return]

VCT- [Return]

BR1- [Return]

BR2- [Return]

DVC-2 [Return]

SR1-40 [Return]

SR2- [Return]

SR3- [Return]

SR4- [Return]

FPBA  DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000003
 SR1-000040 SR2-000000 SR3-000000 SR4-000000

*MDL BMHA [Return]                      ;Enter Module BMHA

DVA- [Return]

VCT- [Return]

BR1- [Return]

BR2- [Return]

DVC-2 [Return]

SR1- [Return]

SR2- [Return]

SR3- [Return]

SR4- [Return]
```

```
BMHA  DVA-000000 VCT-000000 BR1-000000 BR2-000000 DVC-000000
  SR1-000000 SR2-000000 SR3-000000 SR4-000000
*EX [Return]                          ;Leave CNF mode
```

F.3.2.2 Building the C-Table Using the No-Prompt Switch

In this example, DXCL does not prompt you for module entry information.

```
*CNF/NP                  ;Enter CNF mode with prompting
                         ;inhibited
MONITOR: E               ;Enter monitor name
*MDL RMAA[Return]        ;Enter module RMAA
*MDL LPAA[Return]        ;Enter module LPAA
*SR1 77000               ;Change LPAA SR1 value
*MDL TMBA[Return]        ;Enter module TMBA
*DVC 2                   ;Change TMBA DVC value
*SR1 40                  ;Change TMBA SR1 value
*MDL RSAA[Return]        ;Enter module RSAA
**MDL DHAA[Return]       ;Enter module DHAA
*DVA 160200[Return]      ;Change DHAA DVA value
*VCT 300[Return]         ;Change DHAA VCT value
*MDL CPBA[Return]        ;Enter module CPBA
*MDL CPAA[Return]        ;Enter module CPAA
*MDL FPBA[Return]        ;Enter module FPBA
*MDL BMHA[Return]        ;Enter module BMHA
*EX                      ;Leave CNF mode
```

F.3.3 Linking the RTE

You issue the LINK command to build the RTE against a module library, create an executable RTE, and write it to an output device.

```
*LINK DK0:ESAMC0.BIN<DK0:XMOND0.LIB Return  ;Enter the LINK Command

Device not on system

SYS SIZE:160000                    ;Enter System Size

MAKE OUTPUT READY.  WRITE ENABLE

TYPE <CR> WHEN READY.   Return

PASS 1

TRANSFER ADDRESS: 02200

LOW LIMIT: 000000

HIGH LIMIT: 122660

PASS 2

LINK DONE

*SAVC DK0:CSAMC0.CNF               ;Save the configuration table

DONE

*SAVM DK0:MSAMC0.MAP               ;Save the exerciser load map

DONE

*EXIT
```

Appendix G
XXDP Error Messages

XXDP may display error messages while it runs. The following table lists XXDP error messages, the XXDP facility that generates them, and their explanation.

Certain errors are classified as “device errors.” These are specific errors detected by device drivers and communicated to the calling program, which may append additional information to the error report.

? ADDRESS NOT FOUND

Facility: PATCH

Explanation: The specified address does not exist as an entry in the input table.

? BAD ADDR

Facility: Monitor

Explanation: An invalid address was specified with a RUN or START command.

? CHECKSUM ERROR

Facility: Monitor, UPDAT, PATCH

Explanation: Each block in a binary file has a checksum stored in it. This message is printed when the checksum calculated during a block read operation does not match the checksum stored with the block. Try the operation again. The file was probably corrupted.

? COMMAND NEEDS ARGUMENT

Facility: PATCH

Explanation: The command typed by the operator requires an argument but none was given.

? DELETE OLD FILE

Facility: PATCH

Explanation: The specified output file name already exists.

? DEVICE BOOT BLOCK NOT INITIALIZED

Facility: Monitor

Explanation: You have tried to boot a device that does not contain an initialized boot block. The device is not booted.

? device error

Facility: Monitor, UPDAT

Explanation: Each read/write device driver may detect an error during an operation. The driver reports the type of error and returns control to the program being used. The errors that can be reported by drivers are: READ ERROR and WRITE ERROR.

? device error DIRECTORY

Facility: PATCH

Explanation: The error occurred during the operation indicated. Possible device errors are: READ ERROR (error occurred while reading a block of data), WRITE ERROR (error occurred while writing a block of data), and HARD ERROR (error occured during a nontransfer operation).

? device error ON INPUT DEVICE

Facility: Monitor, UPDAT, PATCH

Explanation: The error occurred on the input device specified in the last operator command. If the error persists, the media or the hardware may be bad. Try running diagnostics for the specific device.

? device error ON INPUT DEVICE DIRECTORY

Facility: Monitor, UPDAT

Explanation: The error occurred while accessing the directory on the input device specified in the last operator command. There may be problems with either the device or the media. Try running diagnostics for the device.

? device error ON OUTPUT DEVICE

Facility: Monitor, UPDAT, PATCH

Explanation: The error occurred on the output device specified in the last operator command. If the error persists, the media or the hardware may be bad. Try running diagnostics for the specific device.

? device error ON OUTPUT DEVICE DIRECTORY

Facility: Monitor, UPDAT

Explanation: The error occurred while accessing the directory on the output device specified in the last operator command. There may be problems with either the device or the media. Try running diagnostics for the device.

? device error WHILE LOADING DRIVER FOR dev

Facility: Monitor, UPDAT, PATCH

Explanation: The error occurred while the driver for the specified device was being loaded into memory.

? device error WHILE READING file name

Facility: Monitor, UPDAT, PATCH

Explanation: The error occurred while the specified file was being read.

? DEVICE FULL

Facility: Monitor, UPDAT

Explanation: The capacity of the output device has been exceeded. In the case of random access devices (disks), there are not enough physical blocks remaining to store the file. Blocks allocated during the attempt to write the file are deallocated. In the case of sequential access devices (tape), the physical end-of-tape mark was reached while the file was being written. In both cases, no file is created. Delete some existing files or use another media.

? DIRECTORY FULL

Facility: Monitor, UPDAT

Explanation: There are no remaining entries in the directory and the name of the file and other data cannot be entered. No file is created. Delete some existing files or use another media.

? END-OF-MEDIUM

Facility: PATCH

Explanation: While PATCH was reading a file, the end of the file was encountered before it was expected.

ERR HLT

Facility: DRS

Explanation: A halt-on-error has occurred. DRS returns to command mode. Halt-on-error occurs only if the operator has specified a DRS flag to enable this mode.

? FILE ALREADY EXISTS

Facility: Monitor, PATCH

Explanation: The name of the output file you have specified matches that of a file that already exists on the output media. Delete the old file or use a different name.

? FILE NOT FOUND

Facility: PATCH

Explanation: The specified input file name does not exist.

? FILE TOO BIG

Facility: PATCH

ILL INTER nnn PC nnnnnn PS nnnnnn

Facility: DRS

Explanation: An unexpected interrupt occurred through vector "nnn." The message gives the program counter and processor status word at the time of the interrupt.

? INPUT TABLE EMPTY

Facility: PATCH

Explanation: The specified command cannot be executed, because there are no entries in the input table.

? INPUT TABLE FULL

Facility: PATCH

Explanation: The input table is full and cannot accept any more entries.

INSUFF MEM

Facility: DRS

Explanation: There is not enough memory space to store table information about the number of units that the user wants to specify.

? INVALID ADDRESS

Facility: PATCH

Explanation: An address given in the last command was not legal (possibly an odd number). Check the command and reenter the address.

? INVALID ADDRESS

Facility: UPDAT

Explanation: Three operations can cause this error:

1. You used the MOD command to modify a virtual location but gave an address that was an odd number or that did not fall between the upper and lower core limits.
2. The address given in a LOCORE command was higher than the current high core limit.
3. The address given in a HICORE command was lower than the current low core limit.

? INVALID COMMAND

Facility: Monitor, UPDAT, PATCH

Explanation: You have entered a command that is not recognizable. Check the command (especially spelling) and reenter it.

? INVALID DEVICE

Facility: Monitor, UPDAT, PATCH

Explanation: This error has a number of causes, depending on the command being used. For file-related commands (DIR, COPY, ZERO, SAVE, SAVM, DEL, BOOT, and EOT), one of the devices specified is not file-structured (like paper tape). For EOT, the device is a nontape device. For COPY, the specified devices are not identical types.

? INVALID file name

Facility: Monitor, UPDAT, PATCH

Explanation: You specified a file name in the previous command that was not in the proper format.

? INVALID MODULE NAME

Facility: PATCH

Explanation: A DEC/X11 module name was incorrectly specified.

? INVALID NUMBER

Facility: Monitor, UPDAT, PATCH

Explanation: A number specified in the last command was not correctly entered. Possible problems are: not a number (e.g., 12e4) or not proper radix (e.g., 1292 is not octal).

? INVALID SWITCH

Facility: Monitor, UPDAT

Explanation: The last command was entered with a switch that is not recognized by UPDAT. reenter the command.

INVAL SWTCH FOR CMND

Facility: DRS

Explanation: The user specified a nonexistent or nonapplicable switch in the previous command. See Table 3–2.

INVAL UNIT

Facility: DRS

Explanation: The user specified a unit that does not exist.

? LOGICAL DEVICE NOT ASSIGNED

Facility: Monitor, UPDAT

Explanation: An attempt was made to use a logical unit number without first assigning it to a device. See Section 4.2.1.

LOOKUP ERROR filnam

Facility: DRS

Explanation: The XXDP monitor generates this message. If the file name shown is DRSXM.SYS or DRSSM.SYS, the diagnostic being run requires the DRS, but the corresponding DRS file does not reside on the system media. If the file name shown is another name, the diagnostic attempted to open a file that does not exist on the system media.

LOOP CHNG

Facility: DRS

Explanation: The range of the loop changed while looping on error was in progress.

? MEMORY ERR AT LOCATION: xxxxxxxx

Facility: Monitor

Explanation: A nonexistent memory or a memory parity error occurred at the location specified. If the type of error is detected, it is also reported.

? MODULE NAMES NOT ALLOWED WITHOUT MAP FILE

Facility: PATCH

Explanation: The operator specified a module name in the MOD command without first loading the proper MAP file.

? MODULE NAME NOT FOUND

Facility: PATCH

Explanation: The specified module name does not exist within the DEC/X11 runtime exerciser.

? MUST BE EVEN

Facility: PATCH

Explanation: The operator specified an odd number as an address.

? MUST BE OCTAL

Facility: PATCH

Explanation: The operator specified a nonoctal number.

? NEED NUMBER

Facility: PATCH

Explanation: The operator omitted a numeric value from a command that expected one.

? NO DEVICE DEFAULTS

Facility: PATCH

Explanation: Default device names are not allowed.

NO UNIT

Facility: DRS

Explanation: There are no active units. Either no units have been specified or all units have been dropped.

? NOT ENOUGH ROOM TO LOAD DRIVER

Facility: PATCH

Explanation: The driver for the specified device does not fit into memory.

? NOT FOUND

Facility: UPDAT, PATCH

Explanation: The input file specified was not found on the specified device.

? NOT FOUND: filnam

Facility: Monitor

Explanation: The monitor did not find the specified file. The monitor can read only files that reside on the system media, unless another device is specified.

? NOT FOUND: xx.SYS

Facility: Monitor

Explanation: The driver of device “xx” was not found on the system media. Transfer the required driver file, xx.SYS, to the system media.

? NOT FOUND: xx.SYS

Facility: UPDAT

Explanation: The driver for device "xx" was not found on the system media. This message is printed by the monitor driver that is used to load device drivers for UPDAT. Transfer the required driver file to the system media.

NOT HALTED

Facility: DRS

Explanation: The user issued a PROCEED command, although the DRS had not executed a halt-on-error sequence.

? NUMBER TOO BIG

Facility: PATCH

Explanation: The value typed was too large for its intended purpose.

? OPTION MODULE NAME NOT FOUND

Facility: PATCH

Explanation: The specified option module does not exist.

? OVERFLOW

Facility: Monitor, UPDAT

Explanation: An attempt was made to load too large a program into the program buffer.

PASS ABORTED FOR THIS UNIT

Facility: DRS

Explanation: Testing of the current UUT was prematurely ended. The diagnostic usually displays an error message before DRS displays this message. Refer to the diagnostic's documentation to learn the possible causes of premature test termination.

? READ ERROR

Facility: Monitor

Explanation: A device error occurred while reading. Retry the operation.

? SPECIFY DEVICE

Facility: Monitor, UPDAT, PATCH

Explanation: The command does not allow the use of default device specifications. Reenter the command with the device(s) explicitly specified.

? SYNTAX ERROR

Facility: Monitor, UPDAT, PATCH

Explanation: You incorrectly entered the last command. Reenter it.

TRAP ERR AT: nnnnnn

Facility: DRS

Explanation: The diagnostic executed an unrecognized TRAP instruction. The TRAP instruction is used to communicate between the diagnostic and DRS. This error should never occur in field operation. Please report the problem to Low End Diagnostic Engineering if it does.

TST TOO BIG

Facility: DRS

Explanation: You specified a test number that is larger than the number of tests in the diagnostic program being run.

? UNEXPECTED END-OF-FILE

Facility: Monitor, UPDAT, PATCH

Explanation: The logical end of a file was encountered before it was expected. The file in question is corrupt.

? UNEXPECTED INTERRUPT FROM: xxxxxxxx

Facility: Monitor

Explanation: An unexpected interrupt occurred from the indicated location.

? WRITE ERR

Facility: Monitor

Explanation: A device error occurred while the device was being written to. The device driver may report additional information, which is appended to the message. The user can detect and correct some types of errors, for example, that the device is write-locked or does not have a write ring installed.

? WRONG MAP FILE FOR MONITOR TYPE

Facility: PATCH

Explanation: The MAP file in memory does not have the specified monitor type.

Appendix H
DXCL Messages

DXCL generates keyboard error messages, configure/linker messages, and runtime messages.

H.1 Keyboard Error Messages

The following list gives explanations of the error messages generated by inappropriate entry procedures and RUN commands.

ADDRESS OK BUT EXERCISER WON'T FIT

Explanation: You issued a RUN or RUNL command, but there is not enough room to contain the exerciser between the address specified and the top of memory.

INVALID ADDRESS

Explanation: You entered a nonexistent address, one that is greater than 16 bits, or one that is otherwise not allowed by the monitor.

INVALID COMMAND

Explanation: You entered a command other than those listed in Section A.9.

INVALID COMMAND IN RUN MODE

Explanation: You entered a command while DXCL was in run mode (BSY>), but the command is valid only in command mode.

INVALID MODULE NAME

Explanation: The module name you entered is not 5 characters long or is otherwise unrecognizable by the monitor.

INVALID OR MISSING ARGUMENT

Explanation: You did not fill in all parameters of a command or you supplied one that is not necessary.

MAP BOX MUST BE ON

Explanation: You specified an address parameter greater than 96K (600000) with a RUN or RUNL command, but 22-bit mapping is disabled (MOFF command).

MUST BE EVEN ADDRESS

Explanation: You entered an odd number in an address parameter.

MUST HAVE KT ON

Explanation: You specified an address argument with the RUN or RUNL command, but the memory management unit (KT11) is Off.

NO MODULES SELECTED

Explanation: You entered a RUN or RUNL command, but all modules are deselected.

NOT AN OCTAL NUMBER

Explanation: You entered a nonoctal number (not in the range of 0-7) or you entered an alphabetic character.

NUMBER TOO LARGE

Explanation: You entered a number that exceeds the allowable maximum of 16 bits (that is, 177777 octal).

H.2 Configurator/Linker Error Messages

The following list gives explanations of configurator/linker error messages.

? CHECKSUM ERROR

Explanation: A checksum error occurred during the reading of a binary formatted block.

? CNF TABLE FULL

Explanation: The maximum number of entries (that is, 20 or 40) has been made in the configuration table.

? COR EXCD

Explanation: During the linking process the range of the RTE program exceeded the core size of the system for which it was being generated.

? DEVICE FULL

Explanation: There is no more room available on the specified output device.

? END-OF-MEDIUM

Explanation: The end of an input media has been reached (for example, EOT). This can occur as the result of an unsuccessful block search within a file (for example, EOF).

(ERR01) Symbol Table Error

Explanation: The program detected an error in the symbol table during the linking process.

(ERR02) Global Search Error

Explanation: A global search in the relocation directory (RLD) failed during the linking process.

(ERR03) No PC MOD Command

Explanation: The relocation directory (RLD) does not contain a program counter (PC) modification command. A program error was detected during the linking process.

(ERR04) GSD Block Missing

Explanation: A global symbol directory (GSD) block was not found at the start of the object module. This could be a program error detected during the linking process, or, if magnetic tape is the input media, the tape could have been loaded backwards.

(ERR05) Module Name Missing from GSD

Explanation: The first entry in the global symbol directory (GSD) is not an object module name. This could be a program error detected during the linking process, or, if magnetic tape is the input media, the wrong tape could have been loaded.

(ERR06) Section Name Missing

Explanation: A section name, specified by the relocation directory (RLD), cannot be found; program error.

(ERR07) Can't Find Module Name in Symbol Table

Explanation: A module name is missing from the symbol table, possibly because an option module's file name does not match the name in the header.

(ERR09) Jump Table Index Error

Explanation: The jump table index value exceeds the required range (that is, the GSD code byte is too large); program error.

(ERR12) Load Module Error

Explanation: The program detected an error during the writing of the RTE load module on the output media.

FILNAMEXT ? NON-EXISTENT FILE

Explanation: The file named FILNAM.EXT, specified in the command format, does not exist on the employed media; correct and reenter.

? INVALID COMMAND

Explanation: An invalid command was entered; correct and reenter.

? INVALID NAME

Explanation: An invalid name was used in a command format or as a response to a program request (for example, special characters are not allowed); correct and reenter.

? INVALID SWITCH

Explanation: An invalid switch was used with a command that accommodates switches or a switch was included with a command that does not accommodate switches.

? MUST BE OCTAL

Explanation: The number typed in response to a program request was not octal; reenter octal number.

? NO ROOM FOR A DRIVER

Explanation: The device driver specified in the command cannot be placed in the driver buffer.

? NOT IN CNF MODE

Explanation: A configure mode command (for example, DVA, VCT, BR1, and so on) was entered in nonconfigure mode; type *CNF Return to reenter configure mode and try the command again.

? NUMBER TOO BIG

Explanation: The number typed in response to a program request is larger than allowed for the requested parameter. For example, the device count in the C-table must not exceed decimal 16; vector addresses must not exceed octal 774, and so on; correct and reenter.

? PROGRAM OVERFLOW

Explanation: The block size of the input file is greater than the size of the input buffer.

? READ ERROR

Explanation: An error was encountered while attempting to read from the specified input media.

? SYMBOL TABLE OVERFLOW

Explanation: During pass 1 of the linking process, the symbol table used up all available memory space; use a system with a larger memory.

? USE NEW FILE NAME

Explanation: A file name specified in the command already exists; use another name or delete the old file.

? WRITE ERROR

Explanation: An error was encountered while attempting to write onto the specified output media.

H.3 RTE Runtime Informational Messages

This section lists RTE runtime informational messages.

ASCII Messages

Explanation: The monitor provides each module with an ASCII message capability which can be used to report conditions and statistics.

The following is an example of an ASCII message:

```
(LPAA0 PA xxxxxxxx APC yyyyyy PASS# nnnnn
DATA TRANSFERS:  xxxxxx
SOFT ERRORS:    yyyyyy
HARD ERRORS:    zzzzzz
LP IS OFF LINE
```

where:

- LPAA0 is the module name.
- PA xxxxxxx is its physical address.
- APC yyyyyy is the address of the assembled program counter.
- PASS# nnnnn is the decimal number of completed passes.
- DATA TRANSFERS is the decimal number of I/O transfer.
- SOFT ERRORS is the decimal number of recoverable errors.
- HARD ERRORS is the decimal number of unrecoverable errors.
- LP IS OFF LINE is the line printer status.

END OF PASS

Explanation: This is an optional message. You enable it by setting bit 12 (SR12 = 1). The message indicates that a complete pass through a specific module has been completed.

This message is normally inhibited (SR12 = 0), because its generation may significantly decrease throughput.

At the end of a pass, the specified module reruns, unless the pass is called for by a background module. In this case the monitor executes the next background module.

The following is an example of an end-of-pass message:

```
CPAF0 END PASS #00034.  RUNTIME:  000:11:37 PSTIME:  000:00:37
```

where:

- CPAF0 identifies the module.
- END PASS #00034 defines the decimal number of completed passes.
- RUNTIME/PSTIME define the total run time and pass time in hours, minutes, and seconds (if a system clock is available).

MODULE DROPPED

Explanation: The module itself generates this message:

- Via an END call, following the occurrence of a condition that the module defines as abnormal (for example, no drives available)
- Via the monitor, if the total number of allowable system errors (that is, 4) for the modules is exceeded
- Via the monitor, in conjunction with the setting of software switch register bit 15 (SR15 = 1), following the occurrence of an error (whether acknowledged by printout or not)
- Via the monitor, if software switch register bit 14 is reset (SR14 = 0) and the 20th hard or 40th soft error has occurred (whether acknowledged by the message or not). If bit 14 is set (SR14 = 1), the message will not be printed and the module will not be dropped.

A module that has been dropped cannot be reexecuted unless you reenter command mode by typing Ctrl/C or run mode by issuing the RUN or RUNL command.

The following is an example of a MODULE DROPPED message:

```
CPAF0 DROPPED AT APC xxxxxx
```

where:

- CPAF0 identifies the dropped module.
- APC xxxxxx defines the assembled program counter address (as opposed to the physical address) where the drop occurred.

POWER FAILURE OCCURRED

Explanation: When an RTE is restarted following a power failure, DXCL reactivates the original mode of operation (run or command mode) and prints/displays this message.

RELOCATED TO xxxxxx00

Explanation: If you use the RUN or RUNL command to relocate a program in memory, DXCL generates this message. xxxxxx00 is the 22-bit octal physical address to which the program has been relocated.

H.4 RTE Runtime Error Messages

This section lists RTE runtime error messages.

BAD VECTOR

Explanation: This message indicates that the address pointer is invalid since an interrupt service routine cannot be located. This error does not interfere with the operation of the RTE. However, the module containing the faulty pointer will not output an end-of-pass message and will therefore eventually be dropped if a system clock is available.

The following is an example of the BAD VECTOR message:

```
BAD VECTOR:  200
```

The message means that vector 200 is invalid for the device. You can correct this condition by correcting word 10 of the faulty module's header.

Data Errors

Explanation: Test modules, with the exception of extended I/O modules (IOMODX), report data transfer errors via a data error message (invoked by DATER$).

The following is a model of a data error message:

```
DCAA0  PA xxxxxxxx  APC yyyyyy  PASS# mmmmm  ERR# nnnnn
DATA ERROR CSRA aaaaaa  S/B bbbbbb  WAS wwwwww  WRADR dddddd
RDADR eeeeee
```

where:

- DCAA0 is the name of the failing module.
- PA xxxxxxxx 22-bit the physical address of the DATER$ call.
- APC yyyyyy is the assembled PC address of the DATER$ call.
- PASS# mmmmm is the decimal pass number during which the error occurred.
- ERR# nnnnn is the total decimal error count for the current test run.

- CSRA aaaaaa is the address of the control status register corresponding to the failing device.
- S/B bbbbbb is good expected data.
- WAS wwwwww is bad obtained data.
- WRADR dddddd is the write address of the good expected data.
- RDADR eeeeee is the read address of the bad obtained data.

The address defined by APC locates the DATER$ call that evoked the error message.

Memory Management Errors

Explanation: Aborts and traps generated by the memory management unit (KT11) are vectored through virtual location 250. The memory management status registers (SR0 through SR3) are used to differentiate an abort from a trap, determine why one or the other occurred, and allow for a program restart.

The following is a model of a memory management abort or trap message:

```
*** KT TRAP ***
SR0     SR2
cccccc  cccccc
SR1     SR3
CCCCCC  CCCCCC
```

where:

- SRn identifies a register.
- cccccc gives its contents if available.

Memory Parity Errors

Explanation: Aborts and traps generated by main or cache memory parity errors or main memory ECC errors are vectored through virtual location 114. The control status register (CSR) contains the failure information.

The following is an example of a memory parity or ECC error:

```
**** TRAP THROUGH VECTOR 114 ****
CSR      CONTENTS
aaaaaa    bbbbbb
```

where:

- CSR CONTENTS is the address of the CSR (parity or ECC).
- aaaaaa bbbbb are the CSR contents.

Monitor Data Errors

Explanation: Data transfer errors associated with extended I/O modules (IOMODX) are detected by the monitor via a check data call (CDATA$) request. The data cannot be checked directly, because modules are not mapped contiguously with their write buffers.

A monitor data error message is similar to a data error message with these exceptions.

- All errors detected within a given transfer (for example, a 256-word block) count as a single error (that is, ERR# 00001).
- The count indicated after each error has been reported is a separate message. If SR10 is set, all monitor data errors are reported; if SR10 is cleared, only three errors are reported.
- An additional summary message is provided that defines the total decimal number of errors that have occurred during the transfer.

The following is a model of the monitor check data error message.

```
RKAF0 PA xxxxxxxx APC yyyyyy PASS# nnnnn ERR# nnnnn DATA ERROR
CSRA aaaaaa S/B bbbbbb WAS wwwwww WRADR dddddd RDADR eeeeee
RKAA0 HAD nnnnn ERRORS OUT OF 256 WORDS READ
```

PDP-11/60 and PDP-11/70 Messages

Explanation: If a system error occurs in a PDP-11/60 or 11/70 processor (with an associated DEC/X11 Monitor), related error log messages are output in addition to the standard system error printout previously described.

For a PDP-11/60, the following is included with the system error printout:

```
11/60 ERROR LOG

JAM/xxxxxx SRV/xxxxxx PBA/xxxxxx CUA/xxxxxx
 FLG-INT/xxxxxx WHAMI/xxxxxx CDATA/xxxxxx CTAG-CPU/xxxxxx
```

where:

- JAM/ is the jam register status.
- SRV/ is the service register status.

- PBA/ is the physical bus address register (bits 16,17).
- CUA/ is the microprogram address.
- FLG-INT/ is the flag request register of status/last interrupt vector serviced.
- WHAMI/ are various processor option status bits.
- CDATA/ is the cache memory data word.
- CTAG-CPU/ is the cache memory tag data/hit register.

For a PDP-11/70, the following is included with the system error printout:

```
11/70 ERROR LOG

MEMERREG/xxxxxx CPUERREG/xxxxxx
ADDR/xxxxxxxx
```

where:

- MEMERREG/ is the memory system error register.
- CPUERREG/ is the CPU error register.
- ADDR/ (output only if parity error) is the 22-bit address of the parity error location.

H.5 Soft and Hard Errors

Soft errors are recoverable and hard errors are not. The following is a model of a hard error message:

```
ABCD0  PA xxxxxxxx  APC yyyyyy PASS# nnnnn HARD ERR# nnnnn
CSRA aaaaaa  CSRC cccccc  STATC ssssss  ERRTYP nnnnn
```

where:

- ABCD0 is the name of the failing module.
- PA xxxxxxxx is the actual 22-bit physical address of the error call.
- APC yyyyyy is the assembled PC of the error call.
- PASS# nnnnn is the decimal pass number during which the error occurred.
- HARD ERR# nnnnn is the total decimal number of hard errors encountered.
- CSRA aaaaaa is the address of the control status register (if any) corresponding to the failing device.

- CSRA cccccc are the contents of the device's CSR (if any).
- STATC ssssss are the contents of the device status register (if any).
- ERRTYP nnnnn is the octal code that defines the type of error. (For the error code, refer to the *DEC/X11 Cross-Reference Manual.*)

You use the APC to locate the call that inititated the error.

Extended Soft and Hard Error Printouts

Extended soft and hard error messages contain the same information as soft and hard error messages. They also contain one or more additional lines of information, consisting of up to eight octal values per line. You can find the meaning of the additional data in the context of the specified module in the *DEC/X11 Cross-Reference Manual.*

The following is a model of an extended hard error message:

```
ABCD0  PA xxxxxxxx  APC yyyyyy  PASS# nnnnn HARD ERR# nnnnn
CSRA aaaaaa  CSRC cccccc  STATC ssssss  ERRTYP nnnnn
xxxxxx  xxxxxx  xxxxxx  xxxxxx  xxxxxx  xxxxxx  xxxxxx  xxxxxx
```

**** SYSTEM ERROR ****

Explanation: DXCL outputs a system error message when a bus error trap to location 4 or a reserved instruction trap to location 10 occurs.

The following is a model of a system error message:

```
**** SYSTEM ERROR ****
VECTOR    PC+    ADDR    PSW    SP    ERCT
aaaaaa bbbbbb cccccc dddddd eeeeee ffffff
AT ggggg hhhhhh
```

where:

- aaaaaa is 000004 if bus error trap and 000010 if reserved instruction trap.
- bbbbbb is the program counter address pushed on the stack at the time of failure.
- cccccc is the actual physical address of the error (bbbbbb if there was no relocation).
- dddddd is the processor status word at the time of failure.
- eeeeee are the contents (virtual address) of the stack pointer register at the time of failure.
- ffffff is the decimal system error count.

- ggggg is the module name, if the error occurred within an option module.
- hhhhhh is the assembled program counter (APC) address, if the error occurred in an option module.

Once the system error message has been output, the following occurs:

- The system remains in command mode if the error occurred while it was in command mode.
- The system returns to run mode if the error occurred while it was in run mode or in chain mode. Pass count and error count data are not cleared.

Glossary

This is a glossary of terms commonly used in connection with XXDP.

autodelete
A function of the file transfer process that occurs when a file from the input media automatically replaces a file of the same name on the output media. In UPDAT, only transfers initiated by a FILE command can result in autodeletion.

boot block
The first physical block on a media (block 0). This block contains the XXDP secondary bootstrap for the device.

bootstrap
Simple code used to load and start complex code from a media such as a disk. The term comes from the phrase "picking oneself up by the bootstraps." See also *primary bootstrap* and *secondary bootstrap*.

buffer
A section of memory reserved for storing data, as opposed to executable code. The data usually comes from a file.

console terminal
The video or hard-copy terminal attached to the system via a communications interface at bus address 177560.

device driver
Software that controls the operation of a specific hardware component in a system. An RX02 driver, for example, is software that accomplishes such tasks as selecting a physical block and reading a block of information on an RX02 disk.

device handler
See *device driver*.

dump
The process whereby an image of the contents of memory is placed on a storage media.

edit
Modification of text information in a file.

editor
A utility program used to modify text files.

hardware parameter table
Data structure in which DRS stores hardware information about units under test.

load
The process whereby the program image contained in a file is placed in memory.

media
Physical storage such as a disk or magtape. In this manual, the term "media" is equivalent to "XXDP media."

pass
A unit of diagnostic operation. A DRS-type diagnostic pass is defined as one execution of all specified tests on all active units.

patch
A temporary remedy for a problem in a program that is accomplished by altering the program image stored on the XXDP media.

physical block
A group of data consisting of 256 (decimal) words. This is the standard size of data transmission to and from the XXDP media.

physical location
An absolute memory reference. See *virtual location.*

primary bootstrap
Code, usually stored in a ROM, which loads the boot block (block 0) from a media into the first 256 (decimal) words of memory and then transfers control to memory location 0.

program buffer
A section of memory used by UPDAT for loading program images.

secondary bootstrap

Code that resides in the boot block (block 0) of a media. The primary bootstrap loads and starts this code, which in turn loads and starts the XXDP monitor.

software parameter table

Data structure in which DRS stores information about the operational characteristics of a diagnostic.

switch

A modifier appended to a command.

system media

The media on the device from which the XXDP system was booted.

text

A collection of ASCII-formatted data consisting of printing characters, tabs, carriage returns, and form feeds.

virtual location

A relative memory reference. A program image that has been loaded into the program buffer by UPDAT uses virtual locations—that is, program location 0 is not physical memory location 0 but the first memory location in the program buffer. In contrast, the XXDP monitor performs absolute loads—that is, location 0 is physical memory location 0.

XXDP media

Physical storage, such as a disk pack, magtape, cassette, and so on, that has been formatted for XXDP use.

Index

HOW TO ORDER ADDITIONAL DOCUMENTATION

From	Call	Write
Alaska, Hawaii, or New Hampshire	603–884–6660	Digital Equipment Corporation P.O. Box CS2008 Nashua NH 03061
Rest of U.S.A. and Puerto Rico[1]	800–DIGITAL	

[1]Prepaid orders from Puerto Rico, call Digital's local subsidiary (809–754–7575)

From	Call	Write
Canada	800–267–6219 (for software documentation) 613–592–5111 (for hardware documentation)	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6 Attn: Direct Order Desk
Internal orders (for software documentation)	—	Software Supply Business (SSB) Digital Equipment Corporation Westminster MA 01473
Internal orders (for hardware documentation)	DTN: 234–4323 508–351–4323	Publishing & Circulation Services (P&CS) NRO3–1/W3 Digital Equipment Corporation Northboro MA 01532

Reader's Comments

XXDP User's Manual
AA–FK83A–TC

Your comments and suggestions will help us improve the quality of our future documentation. Please note that this form is for comments on documentation only.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (product works as described)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What I like best about this manual: ______________________________

____

What I like least about this manual: ______________________________

____

My additional comments or suggestions for improving this manual:

____

____

I found the following errors in this manual:

Page	Description
______	______________________________
______	______________________________
______	______________________________

Please indicate the type of user/reader that you most nearly represent:

☐ Administrative Support	☐ Scientist/Engineer
☐ Computer Operator	☐ Software Support
☐ Educator/Trainer	☐ System Manager
☐ Programmer/Analyst	☐ Other (please specify) ______________
☐ Sales	

Name/Title ______________________________ Dept. ______________

Company ______________________________ Date ______________

Mailing Address ______________________________

______________________________ Phone ______________

10/87

— — Do Not Tear — Fold Here and Tape — — — — — — — — — — — — — — — — — —

digital™

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
CORPORATE USER PUBLICATIONS
PKO3-1/30D
129 PARKER STREET
MAYNARD, MA 01754-2198

— — Do Not Tear — Fold Here —

Cut Along Dotted Line