IDENTIFICATION

PRODUCT CODE:	AC-F348G-MC
PRODUCT NAME:	CHQUSGO XXDP/DRS USER S MAN
PRODUCT DATE:	2 JAN 1985
MAINTAINER:	MSD DIAGNOSTIC ENGINEERING
AUTHOR:	MSD DIAGNOSTIC ENGINEERING

PREFACE

Who Should Read This Book

The manual contains both introductory and reference information. Each chapter is divided into description and commands The description section introduces the concepts of the system. The command section describes the commands in alphabetical order.

This manual contains information not published in previous manuals. In addition, the presentation of the material has been improved for better user understanding.

Chapter Summary

This manual is divided into seven chapters:

Each chapter has several examples of actual usage.

The appendixes provide reference information, as follows:

Conventions

There are two conventions used in presenting command formats throughout this manual. First, a field, or item, in a command that is a variable (such as a file name) is denoted by the use of lower case characters. Fields shown in upper case characters must be entered exactly as shown. Second, opt onal fields in a command are enclosed in square ("[]") brackets. XXDP XM commands require only a few characters to un quely dentify a command. The required characters are shown n upper case and the optional characters are shown in lower case.

For example, consider the following:

```
        R   filnam [addr]               XXDP SM
        Run filnam [addr]               XXDP XM
```

In the SM monitor, you type "R" followed by a file name. In the XM monitor, you can type "R", "RU", or "RUN" followed by a file name. The address is optional, as indicated by the square brackets.

Comparison of the XXDP V1.1 and V2 Monitors

The XXDP V2 monitor has the same functionality as the V1.1 monitor and the following enhancements:

- o Improved console handler to provide video terminal support.
- o Improved operator interface so that it is simpler to use.
- o Memory mapping capability, so that the DRS can be moved to memory above 28k words and provide space for large diagnostics.
- o Reporting capability, which allows data and error reports and end of pass information to be saved on the system media.

Compatibility

XXDP V2 required development of a modified Supervisor. All diagnostics that are now compatible with the old DRS will be compatible with the new DRS providing they do not manipulate the MMU or wr te intc the V2 DRS and monitor area.

Further Information

The following manuals provide additional information:

XXDP V2 File Structure, AC-U035A-MC

XXDP V2 Driver Programmer Guide, AC U036A-MC

This manual was prepared by and is maintained by Low End D'agnostic Engineering. Comments and suggestions are welcome. Internal users (DEC employees) should refer to the DEC phone directory l'st:ng for D'agnostic Engineering, PDP-11 Systems. External users should contact their sales representative.

CONTENTS

CHAPTER 1

INTRODUCTION

XXDP V2 's the diagnostic operating system for PDP/LSI-11 systems.

1.1 Components

The XXDP system consists of the following major components:

- o Monitor
- o Diagnostic run time services
- o Utility programs
- o Loadable device drivers

These four components work together to accomplish the system functionality.

1.2 Monitor

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

- o Load and execution
- o Console terminal services

- o Batch control
- o File services for the system medium

The system medium is the storage medium 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.

1.3 Diagnostic Runtime Services

The Diagnostic Runtime Services (DRS) are an extension of the monitor; they provide non-diagnostic function support for certain types of diagnostic programs. These diagnostics are commonly referred to as **supervisor compatible**. Among the functions that the DRS provide are:

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

1.4 Utility Programs

The Utility Programs use the monitor for typing and receiving messages and for loading read/write device drivers required for file operations.

The following utility programs are available:

- o UPDAT
- o PATCH
- o SETUP
- o XTECO
- o DXCL

Documentation for the first four utility programs is provided in this manual. Documentation for DXCL is available separately.

1.5 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.

1.6 Conventions

Two kinds of conventions are used in XXDP, namely: terminal interface conventions and file conventions. The following sections describe these two kinds of conventions.

1.6.1 Terminal Interface Conventions

The console terminal is supported as part of the XXDP monitor. The console terminal driver is a simple, flag-driven handler. Flag-driven means that no interrupts are used and thus unsolicited interrupts do not interfere with diagnostic programs.

The driver makes no distinction between upper and lower case characters. All printing characters are supported, but some characters have special significance in specific situations. The table below lists these special characters. This table briefly describes the function of the ch?racters. Detailed descriptions and examples are left for later sections of this manual.

Char	Use
:	delimit a device specification (e.g., DU0:)
;	start comment line in a batch control file
.	beginning of a three character extension in in a file name
?	"wildcard" character in file name
*	"wildcard" specificationin file name
=	separator between input and output specifications in a command
<	equivalent to "="
/	command switch

In addition, the terminal driver supports the four control characters listed in the table below. Control characters depicted with an [illegible]-arrow are entered by pressing the "CTRL" key and the designated letter at the same time.

Char	Use
↑C	stop current activity and return control to program (testable in batch control file)
↑Z	stop current activity and return control to program (not testable in batch control file)
↑L	delete entire line of input so new line can be typed
↑S	inhibit typing (XOFF)
↑Q	resume typing (XON)
↑X	resume activity after WAIT command in batch control file

1.6.2 File Conventions

XXDP files are specified by a name and extension. The format of a file specification is:

```
file-name [.ext]
```

File-name can be from one to six characters in length and the extension (ext) can be from one to three characters in length. The name and extension are separated by a dot (.). Only alphanumeric characters (A-Z 0-9) can be used and spaces may not be imbedded. Sample file names:

```
UPDAT.BIN
DRSSM.SYS
TEST.1
XMONCO.LIB
```

XXDP allows the use of the wildcard characters "*" and "?" in file specifications. The "?" is used as a substitute for a character and "*" is used as a substitute for a string of characters in a file specification.

For example, if you want a list of all the files with the '.CCC' extension on a medium, 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
XMONCO.*	all files with the name "XMONCO" and any extension
XMONCO.???	same effect as above

1.6.2.1 Extensions

Some extensions are used to identify particular file types. For example, batch control files have ".CCC" extensions. Below is a table of extensions that have particular meanings.

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

THE MONITORS

The XXDP V2 operating system consists of two monitors. The XM monitor and the SM monitor. The XM monitor is used when a memory management unit exists on the system and passes the requirements for booting this monitor. The SM monitor is booted by a chain file user request or in cases when the XM monitor cannot be booted. The SM monitor emulates the capabilities of the XXDP V1 monitor, with some restrictions .

This chapter describes the monitor and then list the commands for each monitor.

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 1.5Kw reserved section is corrupted during diagnostic operations. The monitor does not rely on hardware interrupts for its operation in order to minimize dependency on hardware functionality and impact of malfunctioning equipment.

2.1.1 Monitor Size and Components

At load time the XXDP monitor is about 8K words in size and consists of three major sections: secondary bootstrap, initialization code and the runtime monitor code. The secondary bootstrap is loaded into memory at boot time and 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. The start up process is described in more deta l below.

The runtime monitor consists of five sections:

- o read-write device driver -- loads programs from the system medium and reads batch control files.
- o console terminal driver
- o monitor services handler -- processes requests for monitor services that are made by utility programs via the EMT instruction.
- o operator interface handler -- processes operator commands from the console terminal.
- o batch control handler -- processes batch files from the system medium.

The runtime monitor is approximately 2K words in size. Since older diagnostic programs expect the monitor size to be 1.5K, the monitor s lower .5K 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 kw area at the top of the first 28 kw of memory. XXDP XM auto-boots if the monitor is destroyed, but all conditions may not be fully restored.

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

```
        11/23 s, Micro PDP-11, 11/73

        Unibus: - 11/24, 11/34, 11/35, 11/40, 11/44,
                  11/45, 11/70, 11/84
```

2.1.3 Hardware Requirements

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

Any additional data helpful for determining the cause of the problem is also reported, but XXDP is not intended as a diagnostic so it does not perform rigorous error detection/reporting.

2.1.4 Diagnostic Restrictions

Diagnostics that interfere with the monitor EMT and TRAP vector area can not use system calls. Many SYSMAC diagnostics do this as normal operation when they load and also when they run.

Restriction:	The XXDP SM monitor requires minimum 16K words of memory.
Explanation:	The boot section requires 8kw of memory plus room for the monitor also any diagnostics that require the PDP-11 Diagnostic Supervisor will require this minimum environment.
Restriction:	The integrity of the 1.5kw base monitor or root at the top of the first 28kw memory must be maintained. Failure to do so will require a manual reboot for contiuned system operations.
Explanation:	This root required to restore the monitor.

2.1.5 XXDP System Start-up Procedure

Follow these steps to start up the system:

1. Halt the processor (after making sure that any operating software has been 'gracefully' shut down). Mount/load your XXDP medium. If you are working on a system that has unknown hardware problems, make sure the load device is write disabled.

2. Re-enable and boot the processor.

3. When 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 An example is:

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

   The message in the example above is printed after booting from an RX02 with the XXDP SM monitor on a system with 24K words of memory, using drive 0. The monitor then relocates the runtime monitor code to the top of available memory (up to 28K words in this case, but it could load into extended memory if the XM monitor were booted) and transfers control to this code.

When these steps have been successfully completed, the monitor types a dot (.) to prompt for commands. The commands for the XXDP SM and XM monitors are given later in this chapter.

2.1.6 XXDP Start-up Process

When you boot the load device, the first physical block of data on the medium is loaded into the first 256(10) 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 medium into memory. When the load is complete, the secondary bootstrap passes control to the initialization code and the boot process is complete. If a detectable error occurs during the secondary bootstrap operation, the processor will halt.

If the boot process is successful, the boot performs the following functions:

1. Reads and executes the "BOOT.CCC" chain file and sets up conditions from this file (either SM or QUIET).

2. Sizes memory (up to 124K words) 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

   Reports any errors found

   Informs user if booting SM monitor instead of XM monitor (if not in QUIET mode).

4. Loads and locates the monitor to the top of 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.

5. Identifies the monitor if not in QUIET mode.

6. Starts the monitor.

The BOOT.CCC file is only executable by the boot section of code and only accepts either the SM or QUIET commands. The QUIET command puts the XXDP XM monitor into QUIET mode and starts the SYSTEM.CCC chain file. Only one of these commands can be used.

If the system is too small for the XXDP XM monitor of if the memory management unit is not operating properly, the XXDP SM monitor starts. Since the XXDP SM monitor is not capable of QUIET mode operation, the QUIET command is ignored by this monitor.

The XXDP monitor assumes you are operating with US-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, 0 for 60 Hz and non-zero for 50 Hz. Chapter 4 on UPDAT explains how to modify files, but a brief example is given here.

```
.R UPDAT

LOAD DY0:XXDPSM.SYS
```

```
        MOD 1000

        001000 000000 1

        LOAD DY0:XXDPSM.NEW

        DUMP DY0:XXDP??.NEW
```

The underlined portion in the above example is typed by XXDP. The user has modified the monitor and saved it on the floppy diskette in drive 0, giving the monitor a new extension to prevent deleting the old monitor. The CREATE command can be used to make the device bootable.

2.2 SM Monitor Commands

This sections describes the SM monitor commands. Note that the "FILL command is no longer supported under the XXDP SM monitor.

The commands for the XXDP SM monitor are:

R	run a program
L	load a program
S	start a program
C	run a batch job (chain)
D	list directory of load medium
E	enable alternative drive for system device
H	type help information
TEST	run a batch file called SYSTEM.CCC

Some commands have optional switches, which consist of a single character preceeded by a "/". Switches are used to modify the command function.

J2

2.2.1 Chain Command

The chain command is used to initiate execution of a batch, or chain, file. The file must be on the system medium and must have a .CCC extension. Some batch operations accept switches.

The format of the chain command is:

```
        C filnam[/switches]
```

Chapter 8 of this manual describes batch control in detail.
Example:

```
        C XTEST/RX
```

This command initiates execution of the file XTEST.CCC. In this chain file you can test for the RX condition and execute different sections depending on this switch.

2.2.2 Directory Command

The directory command is used to obtain a list of all the files on the system medium. This list contains five items of information: the entry number, the complete file specification (name and extension), the date the file was created, the length of the file in 256 (decimal) word blocks and the number of the first block in the file. Most files are "linked"; that is, their blocks are not in order on the medium. A few files are contiguous; that is, their blocks are in order on the medium. Contiguous files are noted in the directory by a "C" following the date.

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

The format of the directory command is:

```
        D[/L][/F]
```

There are two optional switches for the directory command. The '/L" switch causes the directory to be printed on a line printer rather than the console terminal. The "/F" switch causes the directory to printed in a short form, which only gives the entry number and file name.

```
                    Directory Long Form
                    -------------------

 ENTRY#   FILNAM.EXT    DATE         LENGTH   START   REVISION

 1        XXDPSM.SYS    02-JUN-79    12       000100  A.0
 2        DU    .SYS    02-JUN-79    5        000120  B.1
 3        DY    .SYS    02-AUG-79    6        000066  A.2

                    Directory Short Form
                    --------------------

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

2.2.3 ENABLE

The enable command is used to designate a different drive as the system device. For example, if you booted the system from drive 0 of an RX02 and later wanted the monitor to use drive 1 as the system device (that is, as the default device), you could do this without re booting the monitor by using the enable command.

This command is valid for multi-drive devices only and effects drives, not controllers.

The format of the command is:

```
        E drive-number
```

An example of the enable command is:

```
        E 1
```

This command enables drive 1 as the system device.

2.2.4 HELP

The help command provides a brief summary of XXDP commands. The contents of a file named "HELP.TXT" are typed/printed and this file must be on the system medium. There is a switch to cause the summary to be printed on a line printer instead of the console terminal.

The format of the command is:

```
        H[/L]
```

Examples of the help command are:

```
        H       type the XXDP command summary on the console.

        H/L     print the XXDP command summary on the line
                  printer.
```

2.2.5 Load Command

The load command is used to load a file into memory. This command can be thought of as the first half of a run command. The program is not started.

As in the run command process, the full file name of the program that was loaded is printed. All restrictions in the run command apply.

The format of the load command is:

```
        L filnam[.ext]
```

Some examples of the load command:

```
        L DIAG          (load DIAG.BI?)
        L ZDJCA2.NEW    (load ZDJCA2.NEW)
```

2.2.6 Run Command

The run command is used to load and start a program that is stored on the load, or system, medium. (Note: the run command is a combination of the Load and Start commands described below.) The program must be an executable file.

The format of the run command is:

```
        R filnam[.ext] [addr]
```

The file name must be a standard XXDP file name (see Chapter 1 If the extension is omitted, a default extension (.BIN or .BIC) is used. If the medium contains a file with the given name and both default extensions, the first file found is used.

After the program is found and loaded, but before the program is started, the full file name is printed to verify the load. This report is useful in determining which of possibly several programs on a medium is being run after a wildcard specification.

The file is started at the transfer address in the file (or at 200 octal in the absence of a transfer address). You can specify a starting address with the run command, if you want the file to start at a different point.

Some examples of the run command:

```
        R UPDAT          (load/start UPDAT.BI?)
        R SAMPLE.XXX     (load/start SAMPLE.XXX)
        R RXDIAG 204     (load/start RXDIAG.BI?  at location 204)
```

Wildcard characters are permitted in the file specification. The first file found that fits the wildcard description is used.

2.2.7 Start Command

The start command is used to start a file that has been previously loaded into core by a load command. No commands should be issued between a load and start command since the program loaded will most likely be overwritten. The purpose of this command sequence is to allow the user to load a program, halt the processor, modify memory contents, restart the monitor and start the program.

The format of the start command is:

```
        S [addr]
```

If you do not give a starting address, the monitor starts the program at the transfer address in the file. The default start ng address for files without specific transfer addresses 's 200 (octal).

Some examples of the start command:

```
        L RXDIAG                    (load RXDIAG.BI?)
        S                           (start at transfer address)

        L RXDIAG                    (load RXDIAG.BI?)
        S 204                       (start at 204)
```

2.3 XM Monitor Commands

This section describes the XM monitor commands. Note that the FILL command is no longer supported under the XXDP XM monitor.

The XXDP XM monitor commands are summarized below:

BOOT	Boot a device
RUN	run a program
LOAD	load a program
START	start a program
COPY	Copy a file or device
CHAIN	run a batch job (chain)
DATE	Set the date or report the date
DELETE	Delete a file
DIRECTORY	list directory of load medium
ENABLE	enable alternative drive for system device
HELP	type help information
INITIALIZE	Initialize a device
PRINT	Print a file on the system line printer
RENAME	Rename a file to a new name
SET	Set device or system parameter
TYPE	Type a file

Some commands have optional switches. See command syntax below. Switches are used to modify the command function.

2.3.1 Command Syntax

The system accepts commands as either: (1) a complete string containing all the information necessary to execute a command or (2) as a partial string. In the latter case the system prompts for the rest of the information.

General syntax for a command is:

```
command[/switch] input-filespec
output-filespec

or

command[/switch]
prompt1? input-filespec
prompt2? output-filespec
```

where:

command	is the command name
/switch	represents a command qualifier that specifies the exact action to be taken.
prompt	represents the keyboard monitor prompt for more information. An appropriate prompt will be printed only if an input or output file of device specification is omited. Not all commands will print prompts. This is a feature of the monitor only. UPDAT does not have this feature.
input-filespec	represents the file on which the action is to be taken.
output-filespec	represents the file that is to receive the results of the operation.

A filespec represents a specific file and the device on which it is stored. Its syntax is:

```
        dev:filnam.typ

        where:

        dev:    represents a device name. If the name is omitted
                then the system device is assumed by default.

        filnam  represents the one- to six-character name of file.
                Wildcard characters are permitted in the file
                specification.

        .typ    represents the one- to three-character file type.
```

2.3.2 Abbreviating Keyboard Commands

Although keyboard commands are all English-language words and therefore easy to use, it can become tedious to type words like PRINTER and INITIALIZE frequently. Abbreviations can be used which are the minumum number of characters that are needed to make the command or option unique. In the following sections the required part of the name is shown in upper case and the optional part in lower case.

2.3.3 BOOT Command

The boot command directs the monitor to boot another XXDP monitor from another XXDP device. This command line is parsed into a format that is acceptable to UPDAT and then control is passed to UPDAT to boot the device or system specified.

The format of the boot command is:

```
        Boot dev:
```

An example:

```
        B DU0:
```

The appropriate device handler must be present on the system device.

2.3.4 CHAIN Command

The chain command is used to initiate execution of a batch, or chain, file. The file must be on the system medium and must have a .CCC extension. Some batch operations accept switches.

The format of the chain command is:

```
        CHAIN filespec [/switches]
```

Chapter 8 of this manual describes batch control in detail.

2.3.5 COPY Command

The copy command performs a variety of file transfer and maintenance operations. The command line is parsed into a format that is acceptable to UPDAT and then control is passed to UPDAT to perform the operation specified.

The format of the copy command is:

```
        COpy [switch-list] input-filespec output-filespec
```

The following switches can be used:

/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 medium in a predetermined section. The root monitor consists of XXDPSM.SYS merged with driver of the output device.

/Files --copies all files from the specified input device onto the specified output device.

/DEVice -- copies a device in image mode to a like device. A copy to a device which is not identical causes the command to abort. Also bad block devices may also fail during execution of this command.

/DElete -- copies a file or files from the specified input device onto the specified output device and automatically delete any file if it already exists.

Example of the use of the copy command:

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

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

2.3.6 DATE Command

The date command lets you inspect or to set the current system date.

The format of the command is:

```
        DAte [ dd-mmm-yy ]
```

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

If the command is given with a date, that date is set. If the command is given without a date, the current date is printed.

```
        DATE 18-MAY-83          Sets the date

        DATE                    Obtains the current date
        18-MAY-83
```

If no date is presently set a default date of 1-JAN-83 is used.

K3

2.3.7 DELETE Command

The delete command deletes the specified files. The name of each file that is deleted is printed and, for tape devices, the tape is rewound after each file is deleted.

The format of this command is:

```
DElete [/NOnames] [/NORewind] filespec
```

- /NOnames -- prevents the printing of the name of each file as it is deleted.
- /NORewind -- prevents a tape drive from rewinding between files when deleting multiple files.

An example of deleting a file:

```
DELETE DK0:ABC.BIN
```

This command deletes the file ABC.BIN from the device DK0.

2.3.8 DIRECTORY Command

The directory command is used to obtain a list of all the files on the system medium. This list contains five items of information: the entry number, the complete file specification (name and extension), the date the file was created, the length of the file in 256 (decimal) word blocks and the number of the first block in the file. Most files are "linked"; that is, their blocks are not in order on the medium. A few files are contiguous; that is, their blocks are in order on the medium. Contiguous files are noted in the directory by a "C" following the date.

When the directory command is given, the monitor parses the command line, loads the UPDAT utility and passes control to UPDAT.

The format of the directory command is:

```
        Directory [/Printer] [/Fast]
```

There are two optional switches for the directory command.

- o /Printer -- causes the directory to be printed on a line printer rather than the console terminal.
- o /Fast -- causes the directory to printed in a short form: entry number and file name.

```
    Directory Long Form
    -------------------

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


    Directory Short Form
    --------------------

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

2.3.9 ENABLE Command

The enable command is used to change the drive that the monitor considers to be the system device. For example, if the user had booted the system from drive 0 of an RX02 and later wanted to have the monitor use drive 1 as the system device (that is, as the default device), he or she could do this without re-booting the monitor by using the enable command. This command is valid for multi-drive devices only and affects drives, not controllers.

The format of the command is:

```
        Enable drive-number
```

There must be valid XXDP V2 media in residence on the specified drive or the system will not allow the command to complete and an invaild device error will be invoked.

2.3.10 HELP Command

The help command is used to obtain a brief summary of XXDP commands. The contents of a file named "HELP.TXT" are typed/printed and this file must be on the system medium. There is a switch to cause the summary to be printed on a line printer instead of the console terminal.

The format of the command is:

```
        Help [/Printer]
```

- /Printer -- lists the help on the printer.

2.3.11 INITIALIZE Command

The initialize command is used to clear and initialize a device directory.

The format of this command is:

```
        Initialize device
```

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

CAUTION

> All data on the medium prior to a this operation is irretrievably lost after the operation. The monitor makes no attempt to determine what is on a medium and will destroy customer data.

A warning is printed whenever this command is invoked, stating which device is involved. The user must then verify that the zero operation is to take place.

An example of the command is:

```
        INI DY1:
```

There is no default for the device. The device must be on line and write-enabled. The following warning is issued after the command is entered:

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

The only answer that confirms the user's intent to carry on is [illegible].

An additional warning message is printed if you specify the system device in this command.

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

If you wish to proceed with the process, you must type a [illegible].

2.3.12 LOAD Command

The load command is used to load a file into memory. This command can be thought of as the first half of a run command. The program is not started. As in the run command process, the full file name of the program that was loaded is printed. All restrictions in the run command apply.

The format of the load command is:

```
        Load  filespec
```

Some examples of the load command:

```
       LOAD DIAG                      (load DIAG.BI?)
       LOAD ZDJCA2.NEW                (load ZDJCA2.NEW)
```

2.3.13 PRINT Command

The PRINT command lists the contents of one or more files on the specified device to the system line printer.

The format of this command is:

```
        Print [/Norewind] filespc
```

The Norewind switch prevents a tape drive from rewinding between files when printing multiple files. If this switch is not given, the tape is rewound after each file is printed.

Some examples of the print command:

```
        PRINT DUO:SYSTEM.CCC

        P N MM1:*.TXT
```

2.3.14 RENAME Command

The rename command is used to change the file specification of an existing file without doing a transfer. The name of the file as recorded in the directory is changed, but there is no movement of data.

The format of the command is:

```
        REname input-filespec output-filspec
```

In the command syntax illustrated above, input-filespec represents the file to be renamed, and output-filspec represents the new name. The device specified in filespec is assumed to be online and write-enabled; device must be the same for both input and output. This command can not be used on a tape device.

Examples of the rename command:

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

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

2.3.15 RUN Command

The run command is used to load and start a program that is stored on the load, or system, medium. (Note: the run command a combination of the load and start commands.) The program must be an executable file. These are only files with .BIN or .BIC extensions to their names (such as, UPDAT.BIN).

The format of the run command is:

```
        Run filespec [addr]
```

The file name must be a standard XXDP file name. The default extension is .BIN or .BIC. If there is a file w'th both extensions on the medium, the first file found will be used.

After the program is found and loaded, but before the program is started, the full file name is printed to verify the load. This is useful in determining which of possibly several programs on a medium is being run after a wildcard spec'fication.

The file 's started at the transfer address in the file (or at 200 octal in the absence of a transfer address). If you want to start at a different address, you can specify the starting address.

Some examples of the run command:

```
        RUN UPDAT           (load/start UPDAT.BI?)
        RUN SAMPLE.XXX      (load/start SAMPLE.XXX)
        RUN RXDIAG 204      (load/start RXDIAG.BI?
                                 at location 204)
```

W'ldcard characters are permitted in the file specification. The first file found that fits the wildcard description will be run.

2.3.16 SET Command

The set command changes device handler characteristics and certa n system configuration parameters.

The format of this command is:

```
        SEt [ physical-device-name ] cond tion
            [ item ]
```

In the command syntax above, "physical-device-name' represents the the device handler whose characteristics need to be modified. The argument "item" refers to a system parameter that needs to be modified. Presently the only system parameter that is changed by the set command is the following:

```
        SET TT:SCOPE
```

Th s option echos RUBOUT characters as backspace-space-backspace The default is NOSCOPE.

```
        SET TT:NOSCOPE
```

This option echos RUBOUT characters by enclosing the deleted characters in backslashes. This is the normal mode. The system returns to this condition on reboot.

```
        SET TT:QUIET
```

The QUIET option prevents the system from echoing lines from the cha n file or from diagnostics that maybe running from a cha n f le (providing the diagnostic has UFD support).

```
        SET TT:NOQUIET
```

The NOQUIET opt on echos lines from chain files or from d agnost cs that are running from a chain file. This is default mode.

2.3.17 START Command

The start command is used to start a file that has been loaded into core by a load command. No commands should be issued between a load and start command since the program loaded will most likely be overwritten. The purpose of this command sequence is to let you load a program, halt the processor, modify memory contents, restart the monitor and start the program.

The format of the start command is:

```
        Start [addr]
```

You can enter a starting address. The monitor starts the program at the transfer address in the file if you don't give a starting address. The default starting address for files without specific transfer addresses is 200 (octal).

Some examples of the start command:

```
LOAD RXDIAG     (load RXDIAG.BI?)
START           (start at transfer address)
LOAD RXDIAG     (load RXDIAG.BI?)
START 204       (start at 204)
```

2.3.18 TYPE Command

The type command prints the contents of a file on the terminal.

The format of the command is:

```
        Type [/Norewind] filespec
```

In the command above the filespec represents the file or files to be typed. Wild cards are accepted in the filespec. The entire command may be on one line or the system can be relied on to prompt for information. The type command prompt is "FILE?"

- o /Norewind -- prevents a tape drive from rewinding between files when typing multiple files. Default is a rewind after each operation.

CHAPTER 3

THE DIAGNOSTIC RUNTIME SERVICES

The Diagnostic Runtime Services (DRS) are the part of the XXDP System that control compatible diagnostic programs. DRS is an extension to the the XXDP Runtime Monitor that is automatically loaded into memory and started when a compatible diagnostic is run. DRS also provides non-test-related services (such as, console terminal support) to these diagnostic programs.

This chapter has five parts:

1. DRS description
2. commands
3. switches, or command modifiers
4. operational flags.
5. table building process

3.1 Description

There are two DRS systems used by XXDP:

- o DRSSM -- a small DRS system
- o DRSXM -- an extended DRS system

When the XXDP SM monitor is booted, the corresponding small DRSSM is loaded for the diagnostic. When the XXDP XM monitor is booted, the larger DRSXM is loaded for the diagnostic.

The DRSSM provides the same capabl:ties that the old DRS did for XXDP V1.1. The DRSXM allows larger diagnostics to run and provides an additional command as described below.

All diagnostic programs that are compatible with DRS share some important common features. Because of these features, they have identical structures, respond to the same general set of commands, report errors in the same way, gather hardware and operational data in the same manner and are therefore easier to use and control from both a user and system point of view.

If you are unsure of which diagnostic programs on a particular medium are DRS-compatible, you can use the SETUP utility to list these diagnostics. Chapter 6 describes SETUP.

3.1.1 DRS Start-up

The start-up procedure for DRS is straightforward. When you ˙ssue an XXDP run command, the diagnostic is loaded and started. The first thing the diagnostic does is to execute an EMT instruction that transfers control back to the XXDP monitor. XXDP then loads DRS from the system medium.

The DRS file is DRS??.SYS where "??" represents the designator for the small DRS (SM) or large DRS (XM). DRS sizes memory using memory management prior to going into command mode. If this hardware has problems, DRS does not start properly. You must run memory management diagnostics ˙f you encounter this problem.

3.1.2 DRS Concepts

You should be aware of several concepts about DRS.

- o CONSOLE COMMANDS - DRS communicates with you through the console terminal. There are eleven commands available for controlling DRS operation. Unlike older-type diagnostics, DRS does not vary its operation based upon starting address or switch registers".

- COMMAND MODIFIERS (SWITCHES) AND FLAGS - You can alter the effects of a particular command by specifying a "switch" when the command is given. For example, unless otherwise specified, most commands will affect all units (devices) that the diagnostic can test. A switch can be used to limit the effect of commands to certain units only.

- UNITS - The diagnostic acts upon specified hardware. Each individual hardware "entity" is referred to as a unit-under-test (UUT) or, most commonly, as a unit. DRS is equipped to handle up to 64 units. You refer to a unit by a number. The first unit is "0". Units are numbered according to the order in which they were specified (see HARDWARE TABLES below).

- HARDWARE PARAMETER TABLES - DRS-compatible diagnostics do not autosize (determine hardware information by performing bus-related tests). You must give the diagnostic the information about the hardware under test that is necessary. This information is stored in a set of tables called "hardware parameter tables". There is one table for each unit to be tested. The specific information required is dependent upon the diagnostic. The diagnostic program prompts the operator for the information it needs for each unit, starting with unit 0. The important concept that you must grasp is the concept of a "table driven diagnostic" in all of the information about a hardware unit is contained in a table specific to that unit.

- SOFTWARE PARAMETER TABLE - There are operational parameters that you can select that affect the way in which a particular diagnostic will function. This information is placed into a table of data called the "software parameter table". This table (for those readers familiar with earlier processor designs) takes the place of the switch register.

- PASS - A pass, or unit of diagnostic operation, is defined to be the execution of all specified tests for all active units-under-test.

- o TEST - DRS diagnostics are divided into independent stuctures called tests. You can run all tests in a diagnostic or select any subset desired.

3.1.3 Error Messages

When a diagnostic detects an error in the device being tested, it calls upon DRS to report the error to the operator. There are three levels of error messages: header, basic and extended. The first message level supplies some 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 given in the header is:

```
        diagnostic name - "ZNAME"
        error type - "HRD"
        error number - "00002"
        unit number - "5"
        test number - "12"
        subtest number - "0"
        location of error call to DRS - "013134"
```

The error number is for identification and is not a running total of the number of errors that have occurred.

The basic error level is used to give a short, simple description of the error. The extended error level is typically used to give supporting information such as register contents at the time of the error. For example:

```
  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 line is the extended message. Error messages are divided into levels in order to give the operator flexibility in determining what portion(s), if any, of the error reports will be displayed or printed.

3.2 Commands

There are eleven commands to the DRSSM and twelve to the DRSXM. These commands are entered in response to the DRS prompt: DR>. The prompt is issued after the DRS is loaded, after all specified diagnostic operations are completed, after a DRS detected error, after a "halt-on-error" sequence and after DRS has been interrupted by a ↑C (CTRL-C). These are tabulated below and described in the remainder of this section. The commands are grouped by related function.

Execution

START	start the diagnostic and initialize
RESTART	start diagnostic and do not initialize
CONTINUE	continue diagnostic at test that was interrupted by a _↑C
PROCEED	continue from an error halt

Data Collection

REDIRECT	redirect error prints and statistics to another unit and/or the line printer

Units Under Test

DROP	deactivate a unit
ADD	activate a unit for testing
DISPLAY	print a list of device information

Time

TIME	set the time of day or display the current time of day.

Flags

FLAGS	print status of all flags
ZFLAGS	reset (clear) all flags

Statistics

PRINT	print statistical information

Exitting

EXIT	return to XXDP monitor

The descriptions below describe the effect of each command.

These effects may be modif'ed by the use of switches that are described in the next section. Familiarize yourself w'th the commands before trying to use the switches. The com ands can be recognized by the DRS from a minimum of three characters; thus the use of the square brackets. That 's, the start command can be entered as 'STA" or 'STAR" or 'START .

3.2.1 ADD Command

The add command is used to activate a unit for testing. The unit switch is used to specify the unit to be activated (see section 2). All units are initially active and must be explicitly deactivated by the user or the diagnostic. The units to be activated must have already been deactivated.

The format of the command is:

```
ADD [/UNIts:unit-number]
```

unit number is the unit to be activated. Section 2 of this chapter describes the unit switch in detail.

The default operation of the add command is:

- o If the /UNITS switch is specified, the given unit is activated.

- o If the switch is not specified, all deactivated units are returned to active testing.

3.2.2 CONTINUE Command

The continue command is used to resume diagnostic operation after interrupted by a ↑C or after a halt on error. The diagnostic is restarted at the beginning of the test that was interrupted, not at the first test, as would be the case with the RESTART command. The unit being tested when the diagnostic was interrupted remains as the unit being tested. You are given the opportunity to change the software table if you wish. You are not able to change the hardware tables.

The format of the command is:

```
        CONtinue [switch-list]
```

Switch-list is any valid combination of switches (modifiers) for the continue command. Section 2 of this chapter describes these switches.

The default operation of the continue command is:

- The testing runs for the number of passes remaining in the pass count specified in the last start or restart command. (A pass is defined to be all specified units tested once by all specified tests.)

- All flags will remain set/clear as previously specified.

Example of continue command (underlined portions typed by DRS):

```
        ↑C
        DR>CON
        CHANGE SW (L) ?  N
```

You can also use the start and restart commands to resume diagnostic execution, but diagnostic initialization will take place and testing will start with the first unit, first test.

3.2.3 DISPLAY Command

The display command is used to examine the contents of the hardware tables. All table data for the specified units are listed on the console terminal. Units that have been dropped are so designated.

The format of the command is:

```
        DISplay [switch-list]
```

Switch list is any valid combination of switches (modifiers) for the display command. Switches are described in section 2 of this chapter.

The default operation of the display command is:

- All units described in the hardware tables will be displayed on the console terminal.

3.2.4 DROP Command

The drop command is used to deactivate a unit from testing. The unit to be deactivated must be specified using the unit switch (see section 2). All units are initially active.

The format of the command is:

```
        DROp [/UNIts:unit-number]
```

Unit-number is the number of the unit to be deactivated. The unit switch `s described in detail in section 2 of this chapter.

The default operation of the DROP command is:

- If the /UNITS switch is specified, the given unit `s deactivated.
- If the switch is not specified, all active units are dropped from active testing.

3.2.5 FLAGS Command

The flags command is used to find the current status of the DRS flags. Upon receipt of this command, DRS will display the status of all flags on the console terminal.

The format of the command is:

```
        FLAgs
```

Example of the flags command (underlined portion typed by DRS):

```
        DR>FLA
        FLAGS SET
        NONE

        No flags are set.
        DR>FLA
        FLAGS SET
        IER
        LOE
```

There are two flags that have been set: IER and LOE. (These flags are described in a later section.)

3.2.6 PROCEED Command

The proceed command is used exclusively with the halt-on error feature in DRS. When halt on-error s in force and the diagnostic reports an error to DRS, DRS returns to command mode. You can issue any commands at this point. The proceed command is special in that it restarts the diagnostic at the point where it reported the error. No initialization is done, the unit-under-test is not accessed and the vector space is unchanged. This process allows you to examine the state of the unit being tested and then continue testing without disturbing diagnostic operation.

The format of the command is:

```
PROceed [switch-list]
```

Switch-list is any valid combination of switches (modifiers) for the PROCEED command. These switches are described in section 2 of this chapter.

The default operation of the proceed command is:

- o the flags (section 3 of this chapter) remain set/clear as specified with the previous command.

3.2.7 REDIRECT Command

The redirect command is available in the DRSX which operates from extended memory. This command causes certain information to be redirected to a device instead of being printed on the terminal. The information that is redirected is error information, statistics, drop units, and End of Pass information.

The format of this command is:

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

The /DEV switch allows the name of a device to be entered on which the data is to be stored. The data is collected in a file named COLECT.DAT. If this file is not present, it is created and data is added to it. If the file is present, it is opened and data is appended to the end of the file. The file is opened at the beginning of each pass of the diagnostic program and closed at the end of the pass.

The /LPT switch causes the information to be printed on the system line printer in addition to being collected on the device specified by the DEV switch. The file is closed any time there is a return to command level. A ↑C or trap causes a return to command level. The date of the file is the current date and is updated if necessary.

The command can be executed any time at command level. If the redirect is canceled, the above information is output to the console terminal. The redirect command can be canceled by entering the command without any arguments.

Example:

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

Either or both switches can be used in any order. The device driver specified in the DEV: switch must be on the system media. An error message is displayed on the system console if the attempt to load the driver fails.

The information saved on the storage device/serial line or line printer is information normally printed from:

1. Error calls, Error block call, and all PRINT? calls that are within the scope of that error call (for example: the error sub-routine, extended error message, etc.)

2. PRINTS calls

3. the end of pass printout.

Information contained in PRINT? calls that are not within the scope of the error are not saved and are output to the console terminal.

This data is stored in ASCII form just as the diagnostic program passed them to the DRS. No data compression or other formatting is accomplished on this information.

3.2.8 RESTART Command

The restart command, like the start command, starts the diagnostic from an initial state. The diagnostic initialization process may be different in response to a restart. Please refer to diagnostic documentation for details. The vector space is not changed. You only have the opportunity to change the contents of the software table.

The format of the command is:

```
        REStart [switch-list]
```

Switch-list is any valid combination of switches (modifiers) for the restart command. Section 2 of this chapter explains switches.

The default operation of the restart command is:

- o All tests are run on all units.
- o Flags (section 3 of this chapter) are cleared.
- o The testing continues until interrupted by a C or by a system error.
- o An end-of-pass message is printed after each pass. (A pass is defined to all specified units tested once by all specified tests.)

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

3.2.9 START Command

The start command starts the diagnostic from its initial state and should be the first command issued to DRS. All initialization code is executed. Refer to specific diagnostic documentation for exact nature of the initialization process carried out by a particular diagnostic. The "trap catcher" code is reloaded into the vector space. (The trap catcher is code that allows DRS to handle and report any unexpected interrupts.)

The format of the command is:

```
STArt [switch-list]
```

Switch-list is any valid combination of switches (modifiers) for the START command. The switches are explained in section 2 of this chapter.

The default operation of the START command is:

- o All tests are run on all units.
- o All flags (section 3 of this chapter) are cleared.
- o The testing continues until interrupted by a ↑C or by a system error.
- o An end-of-pass message is printed after each pass. (A pass is defined to be all specified units tested once by all specified tests.)

After you issue the start command, you are asked if you wish to change the hardware information. You must answer yes ("Y") to this question if there are no existing hardware tables. Hardware tables will already exist if they were entered:

1. By a previous start command sequence
2. By use of the SETUP utility
3. By a programmer who hardcoded tables into the diagnostic image.

You can override existing tables at this point if you wish.

You are then asked for the number of units to be tested. Enter the decimal number of units. You are asked for hardware-specific information for each unit according to the design of the diagnostic.

Example of START (underlined portions typed by DRS):

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

        [answer diagnostic questions]
        CHANGE SW (L) ?  N
```

You are asked for the hardware data for "n" units, where "n" is a decimal number between 1 and 64. Refer to section 4 of this chapter for assistance in answering these questions. The questions should be obvious and straightforward. If you have difficulty with the questions of a specific diagnostic, please refer to the document for that diagnostic or direct questions to Diagnostic Engineering.

After you enter all hardware data, you are asked if you wish to change the operational data (software table). This is purely optional. You do not have to answer any software data questions unless you want to modify default diagnostic operational behavior. Section 4 of this chapter will assist you in answering the questions, but please refer to diagnostic documentation for explanations of specific questions.

If there are no hardware tables, you get an error message. The following example shows what happens when hardware tables are not present:

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

3.2.10 TIME Command

The time command sets the time of day or displays the current time of day. This command is only valid for DRSXM.SYS.

The format of the command is:

```
        TIME [hh:mm:ss]

    hh  represents the hour    (a decimal number from 1 to 23)
    mm  represents the minutes (a decimal number from 0 to 59)
    ss  represents the seconds (a decimal number from 0 to 59).
```

An example of setting the time:

```
        DR>TIME 11:15
```

An example of obtaining the time:

```
        DR>TIME
        11:15:01
```

If no time is presently set a default time of 00:00:00 will be used and the system will increment from that time. If no clock is available on the system an error message is printed when the time command is invoked.

3.2.11 ZFLAGS Command

The zflags command resets all DRS flags to their cleared state.

The format of the command is:

```
        ZFLags
```

3.3 Switches

Switches are modifiers of command functions. For example, many DRS commands affect units. Usually a command of this type affects all units specified during hardware table build. A switch enables you to limit the effect of the command to certain selected units.

The DRS switches are:

/TESTS:test-list	execute only the tests specified
/PASS:ddddd	execute ddddd passes (ddddd = 1 to 65536)
/FLAGS:flag-list	set specified flags
/EOP:ddddd	report end-of-pass after each ddddd passes (ddddd = 1 to 65536)
/UNITS:unit-list	command will affect only specified units

All switches cannot be used with all commands. The following table shows which commands each switch may be used with.

	TESTS	PASS	FLAGS	EOP	UNITS
ADD					X
CONTINUE		X	X	X	
DISPLAY					X
DROP					X
EXIT					
FLAGS					
PRINT					
PROCEED			X		
REDIRECT					
RESTART	X	X	X	X	X
START	X	X	X	X	X
ZFLAGS					

3.3.1 /EOP Switch

The /EOP switch is used to specify when end-of-pass messages will be printed. These messages indicate the number of passes completed and the number of errors found. Default DRS operation is to print these messages after every pass.

The format of this switch is:

```
        /EOP : number-passes
```

Number-passes is a decimal number between 1 and 65536. The end-of-pass message is printed each time the number of passes specified is completed.

In the example below, the message is printed after every 90 passes. (The underlined portion is typed by DRS.)

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

3.3.2 /FLAGS Switch

The /FLAGS switch is used to set DRS operational flags. These flags are described in detail in the next section of this chapter. Default DRS operation is all flags cleared.

The format of this switch is:

```
         /FLAgs : flag-list
```

Flag-list is a list of DRS flags separated by colons. Please refer to section 3 of this chapter for detailed descriptions of flags.

Some examples of the FLAGS switch (underlined portion typed by DRS):

```
        DR>STA/FLAGS:LOE
        DR>RES/FLA:LOE:IER:BOE
```

3.3.3 /PASS Switch

The /PASS switch is used to specify the number of passes that a diagnostic will run. A pass is all specified tests on all active units. Default DRS operation is "no limit" on passes. This switch allows the you to place a limit on the number of passes.

The format of the switch is:

```
         /PASs : number-passes
```

Number-passes is a decimal number between 1 and 65536.

Some examples of the /PASS switch (the underlined portion is typed by DRS):

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

3.3.4 /TESTS Switch

The /TESTS switch is used to specify what tests will be run. The default DRS operation is to run all tests, but this switch allows you to override the default.

The format of the switch is:

```
         /TESts:test-list
```

Test-list is a list of test numbers separated by colons. If the test numbers are sequential, they can be specified by the first and last test number separated by a dash.

For example, if tests 1, 2, 3 and 4 are to be specified, they may be entered as "1:2:3:4" or "1-4". Test numbers may be entered in any order, but tests will always be executed in numeric order.

Some examples of the /TESTS switch follow. The underlined portion is typed by DRS.

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

In the first command, the user selected test 5 only. In the second command, the user selected tests 1 and 2. In the final command, the user selected tests 1, 5, 6, 7, 8, 9 and 15.

3.3.5 /UNITS Switch

The /UNITS switch is used to specify which available units are to be tested. Default DRS operation is to encompass all units in the scope of any command. This switch is used to limit the effects of a command to certain units. The format of the command is:

/UNIts : units-list

Units list is a list of unit numbers separated by commas. 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. If the units are sequential, they may be specified by the first and last unit number separated by a dash ("-"). For example, units 3, 4, 5, 6 and 7 may be specified as "3-7".

Some examples (underlined portions typed by DRS):

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

The first example drops unit 1. The second example adds units 2 and 3. And, in the last example, the diagnostic restarts with units 5, 6, 7, 8 and 9 being tested.

3.3.6 Combining Switches

You can specify as many valid switches, in any order, with a command as you wish. Simply string out the switches, one after another, on the command line.

For example, if you want to start a diagnostic and:

1. test units 1 through 4 only
2. execute tests 1, 5 and 15
3. execute 100 passes
4. only report the end-of-pass data after every 10 passes

3.4 Flags

Flags are used to set up certain operational parameters such as looping on error. All flags are cleared at startup and remain cleared until explicitly set using the /FLAGS switch. Flags are also cleared after a start or restart command unless set using the /FLAG switch. The zflags command can also be used to clear all flags. No other commands affect the state of the flags.

Flag	Effect
ADR	execute autodrop code
BOE	"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)
IDR	inhibit program dropping of units
IER	inhibit all error reports
ISR	inhibit statistical reports (does not apply to diagnostics which do not support statistical reporting)
IXE	inhibit extended error reports (those called by PRINTX macro's)
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)

The ADR flag, when set, causes DRS to execute the "autodrop" code in a diagnostic. The purpose of this code is to test for 'device ready" or "device available". If the unit being tested is not ready or available, it is dropped (deselected). Not all diagnostics have autodrop code. Refer to specific d agnostic documentation to determine if a d agnostic does support this feature.

3.4.2 BOE Flag (Bell On Error)

The BOE flag, when set, causes DRS to issue a "CTRL-G", or bell' character when an error is reported by the d agnost c. Th s gives an audible tone at the console term nal. Th s feature s normally used in conjunction with

H6

the message inhibit functions.

3.4.3 EVL Flag (EVaLuate)

The EVL flag, when set, causes DRS to execute diagnostic evaluation code. This is an optional feature and you must refer to specific diagnostic documentation.

3.4.4 HOE Flag (Halt On Error)

The HOE flag, when set, causes DRS to execute a "halt-on-error" sequence when an error is detected by the diagnostic. Execution of this sequence does not result in an actual processor halt, but returns DRS to command mode. The exact process is:

1. When the error is reported to DRS, the error message(s) are printed (unless printing has been inhibited).

2. DRS returns to command mode.

3. The diagnostic is suspended at the point of the error report to DRS and the unit being tested is left in the state that it was in at the time of the call.

After DRS has returned to command mode, you can issue a proceed command to resume diagnostic execution at the point where it was suspended. You can also issue other commands as desired.

3.4.5 IBE Flag (Inhibit Basic Errors)

There are three levels of messages in an error report. The IBE flag, when set, causes DRS to inhibit the second and third level of error reports. The first level, which contains the error type, number, pc, test, and unit, is not inhibited.

3.4.6 IDR Flag (Inhibit DRopping of units)

The IDR flag, when set, causes DRS to inhibit the dropping (deselection) of units by a diagnostic. Diagnostics can deselect a unit from the test process if an error threshold 's reached or if a serious error is detected. This flag allows you to keep the unit selected, usually for the purposes of tracing the error.

3.4.7 IER Flag (Inhibit Error Reports)

The IER flag, when set, causes DRS to inhibit all error reporting to the console terminal. While in effect, no messages are sent to the operator except system error reports such as ILL INT (illegal interrupt) and end-of-pass reports. This feature is normally used in conjunction with error looping. It speeds up the test process and, in the case of hard copy terminals, saves paper.

3.4.8 ISR Flag (Inhibit Statistical Reports)

The ISR flag, when set, causes DRS to inhibit the printing of statistics by the diagnostic. This is an optional feature and not al! diagnostics support statistics. Consult specific diagnostic documentation to determine whether or not a diagnostic has this feature.

3.4.9 IXE Flag (Inhibit eXtended Errors)

The IXE flag, when set, causes DRS to inhibit the extended error reporting only. The error reports produced by the PRINTX call are inhibited. The error message and basic reports are printed.

3.4.10 LOE Flag (Loop On Error)

The LOE flag, when set, enables DRS error looping. When error looping is in effect, DRS causes the diagnostic to continually re-execute the code that detected the error. Looping remains in effect even if the symptoms that prompted the error report d'sappear. This allows for looping on intermittent errors. To stop the looping, you must type CTRL-C (C) to return DRS to command mode.

3.4.11 LOT Flag (Loop On Test)

The LOT flag, when set, causes DRS to continually execute the test(s) specified with the TEST switch. The initialize and end-of-pass code are not executed as in normal operation however.

3.4.12 PNT Flag (Print Number of Test)

The PNT flag, when set, causes DRS to print the number of the test being executed.

3.4.13 PRI Flag (PRInter)

The PRI flag, when set, causes DRS to redirect all messages to a line printer. This does not apply to command prompts.

3.4.14 UAM Flag (UnAttended Mode)

The UAM flag, when set, prevents the use of manual intervention during testing. Manual intervention assumes that an operator is present to undertake any necessary action. The use of this flag allows the operator to start the diagnostic and let it run unattended. When this flag is in effect, some testing will be inhibited. Refer to specific diagnostic documentation for a description of UAM flag effects in specific cases.

3.5 Table Building

DRS uses hardware tables for unit information (such as register addresses, drive numbers or interrupt priority). Tables are also used for diagnostic-specific operational information (such as what data patterns to use for testing or whether or not to do read-only testing). The specific information varies from diagnostic to diagnostic, so this section only seeks to provide you with some background information on these tables.

These tables must be constructed. They are constructed in three ways.

1. The diagnostic is typically released (distributed) with only a "template" table for hardware data. This template contains default values for hardware information in some cases. In any event, you must build the actual tables. This is most often done by starting the diagnostic with the start and specifying the hardware data as requested by the diagnostic.

2. The table may also be "prebuilt" using the SETUP utility (Chapter 6). SETUP is an XXDP utility program that allows the user to build tables without actually running the diagnostic. The tables are stored with the diagnostic on the XXDP medium and are brought into memory with the diagnostic at runtime. The user may then initiate diagnostic operation without building tables.

3. The tables may have been built by the diagnostic programmer. These tables are already a part of the program image and can be used as they are or changed after a START command.

The operational table, which is called the software (SW) table, may not be present in all diagnostics. If it is not present, you are not asked if you wish to change it. This is an actual storage area that has default data coded into it.

All table-related questions have the same format:

```
        Question (type) [default] ?
```

The question may be something like "DRIVE NUMBER". The type is a one character code for the type of answer desired, enclosed in parenthesis. The possible types and codes are: O for octal, D for decimal, B for binary, A for ASCII and L for logical (Y or N). The question mark indicates that DRS is ready to accept the answer. If the answer is unacceptable for any reason, an error message will be typed and you are asked for the information again.

When you answer the hardware questions, you are building entries in a table that describes the devices under test. The simplest way to build this table is to answer all questions for each unit to be tested. If you have a multiplexed device such as a mass storage controller with several drives or a communication device with several lines, this becomes tedious since most of the answers are repetitious.

To illustrate a more efficient method, suppose you are testing a fictional device, the XY11. Suppose this device consists of a control module with eight units (sub-devices) attached to it. These units are described by the octal numbers 0 through 7. There is one hardware parameter that can vary among units called the "Q-factor". This Q-factor may be 0 or 1. Below is a simple way to build a table for one XY11 with eight units. The underlined portions are typed by DRS.

```
    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 ?
```

Notice that the default value for the Q-factor changes when a non-default response is given. Be careful when specifying multiple units!

As you can see from the above example, the hardware parameters do not vary significantly from unit to unit. The procedure shown is not very efficient. The runtime services can take multiple unit specifications however. Let's build the same table using the multiple specification feature.

```
            UNITS (O) ?  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
```

As you can see in the above dialogue, the runtime services will build as many entries as it can with the information given in any one pass through the questions. In the first pass, two entries are built since two sub-devices and Q-factors were specified. The services assume that the csr address is 160000 for both since it was specified only once. In the second pass, four entries were built. This is because four sub-devices were specified. The "-" construct tells the runtime services to increment the data from the first number to the second. In this case, sub-devices 2, 3, 4 and 5 were specified. (If the sub-device were specified by addresses, the increment would be by 2 since addresses must be on an even boundary.) The csr addresses and Q-factors for the four entries are assumed to be 160000 and 0 respectively since they were only specified once. The last two units are specified in the third pass.

The whole process could have been accomplished in one pass as shown below.

```
          UNITS (D) ? 8

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

As you can see from this example, null replies (commas enclosing a null field) tell the runtime services to repeat the last reply.

CHAPTER 4

UPDAT UTILITY

UPDAT is a file manipulation utility program used for building XXDP media, copying files from one medium to another, deleting files from a medium, modifying files and other functions. The component name for UPDAT is "CHUP2?? UPDAT UTIL", but for the benefit of the user, the program is released under its common name: UPDAT

4.1 Description

UPDAT runs in the lower part of memory and occupies about 6K words. It uses the runtime monitor for interfacing to the operator and loading the retrievable device drivers that it uses to accomplish device related functions. Since UPDAT requires these device drivers, the drivers you intend to use must be resident on the system medium and the system medium must be available (on-line) throughout your use of UPDAT. There is one exception that is explained later in this chapter in conjunction with the DRIVER command.

4.1.1 Starting UPDAT

To start UPDAT, type:

```
        R UPDAT
```

When the program has been successfully loaded by the monitor, it types its name and a restart address and then types the prompt, * to tell you that it it is ready to accept commands.

4.1.2 Commands Functions

This section describes the command categories and then gives a detailed description of each command. UPDAT commands can be divided into the following categories:

- o File manipulation commands
- o File modification commands
- o New medium creation commands
- o Miscellaneous commands
- o Return to monitor commands
- o Print commands

The following sections describe the commands in each category.

4.1.2.1 File Manipulation Commands

The file manipulation commands are used to maintain XXDP media. Files can be transferred from medium to medium, deleted from a medium or renamed. A directory, or list, of files on a medium may be obtained. The file manipulation commands are:

DIR	give directory of specified medium
PIP	transfer a file or files
FILE	transfer a file or files
DEL	delete a file or files
REN	rename a file

4.1.2.2 File Modification

An important function of UPDAT is the modification of binary files. When a diagnostic program is found to have a deficiency, one of the corrective measures taken is to issue a DEPO or 'patch order'. This is a temporary change to a released program.

UPDAT is one of the means of implementing these temporary remedies. The program in question is loaded from an XXDP medium into an area in memory called the program buffer. This area lies in the physical memory space between the monitor and UPDAT and its size is determined by the amount of memory in the system. The size is equal to the system size minus 8K words, but no larger than 20K words. The program image, now resident in memory, may be modified and then put back onto an XXDP medium ("dumped"). The transfer address and load image size may also be altered by the user at this time.

The next seven commands relate to this function of UPDAT. In the descriptions of these commands, locations within a program that has been loaded into the program buffer are referred to as "virtual locations" since their addresses are relative to the first physical location in the program buffer and not the first physical memory location as would be the case if the program had been loaded by the monitor.

The location addresses given in a DEPO (Diagnostic Engineering Patch Order) are treated as virtual when using UPDAT. For example, if the DEPO says to modify location 1002, that will be the virtual location you will refer to in the MOD command. If the program has been loaded by the monitor however, 1002 would be an absolute address in memory, not a relative location in the program buffer.

The file modification process applies to image (BIC or BIN) files only. The process, briefly, consists of the following steps:

1. Load the file into the program buffer with the LOAD command.

2. Change the size of the image, if necessary, with the HICORE and LOCORE commands.

3. Modify the contents of the desired location(s) with the MOD command.

4. Modify the transfer address, if desired, with the XFR command.

5. Write the image onto media with the DUMP command.

The file modification commands are:

CLR	clear UPDAT program buffer
LOAD	load a program
MOD	modify file image in memory
XFR	set transfer address
HICORE	set upper memory limit for dump
LOCORE	set lower memory limit for dump
DUMP	dump a program image

4.1.2.3 New Medium Creation

The folowing commands are used to create new XXDP media. The build process is described in detail in Appendix E of this manual.

The new medium creation commands are:

ZERO	initialize a medium
CREATE	save a monitor on a disk or tape
COPY	copy entire medium

4.1.2.4 Miscellaneous Commands

The following are miscellaneous commands:

ASG	assign a logical name to a device
DO	execute an indirect command file
READ	read a file to check validity
EOT	write logical end-of-tape mark on a tape
DRIVER	load a device driver

4.1.2.5 Returning to Monitor Commands

The following two commands allow you to return control to the monitor:

BOOT	bootstrap a device
EXIT	return control to the runtime monitor

4.1.2.6 Printing Commands

There are two commands to output textual information from files.

PRINT	print a file on the line printer
TYPE	type a file on the console terminal

4.2 Commands

The detailed descriptions of UPDAT commands follow, in alphabetical order.

4.2.1 ASG Command

The assign command is used to assign a logical unit number to a device. The device can then be referenced by this number in ensuing UPDAT commands.

The format of the command is:

```
        ASG dev:=n


        dev - device to be assigned

        n - logical unit number (0-7)
```

The primary use for the assign command is to facilitate the use of the DO command and indirect command files (see next section).

Example of assign command:

```
        ASG DY0:=0
        PIP 0:=DY1:*.CCC
        FILE 0:=MM0:FILE.NEW
```

4.2.2 BOOT Command

The bootstrap command is used to start the monitor in the same manner as the hardware bootstrap. The purpose of this command is to allow you to boot a device other than the original system device. The booted device is now the system device.

The format of the command is:

```
        BOOT dev:
```

The device must have a bootable medium mounted. The boot process consists of loading the first physical block (boot block) into the first 256 (decimal) words of memory and starting execution at location 0.

4.2.3 CLR Command

The clear command clears the program buffer into which programs are loaded for modification. This command allows the user to assure that unused locations in a program are set to zero when the program image is dumped to a medium.

The format of the command is:

```
        CLR
```

4.2.4 COPY Command

The copy command is used to copy the entire contents of one medium to another identical medium (e.g.; RP06 to RP06). The copy process can take two forms. The first is an "image copy". This is a block-for-block transfer and is very fast since all available memory is used as a buffer. If the device is a bad block device, there is a chance that a bad sector may be encountered during the copy process. In this case the process is aborted. You then have to use the second form, "file copy". This is slower since only one block is transferred at a time. In both cases, the former contents of the medium are destroyed. There is no check for medium type and customer data could be lost. Be careful! There will be a warning message as shown below. The copy will proceed only if a ""Y"" is typed.

The format of the command is:

```
        COPY devo:=devi:


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

where devo and devi are the output and input devices respectively. The two devices must be the same type. Both dev'ces must be onl'ne and the output device must be write-enabled.

4.2.5 CREATE Command

The create command is used to place a bootable monitor on a medium. The command places the appropriate secondary bootstrap on the boot block and places the monitor file on the medium in a predetermined section. Refer to Appendix E for details on building XXDP media.

The medium need not be initialized with the zero command if the medium is already XXDP compatible. In this case only the monitor and bootstrap on the medium prior to the CREATE operation are lost. All other files are preserved.

The format of the create command is:

```
        CREATE dev:
```

dev: is the device with the medium that is to be made bootable. The device must be online and write-enabled. The file XXDPSM.SYS and the appropriate driver must be on the system device to build the monitor boot block image.

4.2.6 DELETE Command

The DELETE command is used to remove a file, or files, from an XXDP medium. The actual process for deletion is to remove the file name from the directory and deallocate the physical blocks used by the file.

The format of the command is:

```
        DEL dev:ifile[/N][/Q]
```

```
  dev - device where file resides; device is assumed to be
        online and write-enabled; there is no default

  ifile file(s) to be deleted; file must be on device
        specified; extension must be specified (unless
        file has no extension); wildcards are accepted

  /N - inhibit printing of file names as they are deleted;
        if switch is not specified, names will be printed

  /Q - do not rewind before searching for file(s) (tape
        devices only); if switch is not specified, tape
        will be rewound prior to searching for each file
        to be deleted
```

Examples of the delete command:

```
        DEL MM0:XYZ001.TXT/Q
```

This command deletes the file named "XYZ001.TXT" on MM0. The search for the file begins at the point where the tape is currently positioned (the no rewind switch has been specified). An error is reported if the device is not online and write-enabled or if the file does not exist.

```
        DEL *.OLD
```

This command deletes all files on the system device with "OLD" extensions. An error is reported if the device is not online and write-enabled or if there are no files with "OLD" extensions.

4.2.7 DIR Command

The DIR command is used to obtain a directory, or list, of files on a specified medium. You can specify where the directory is to go (console terminal, line printer or file) and what form the directory is to take (long or short).

The format of the command is:

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

devo - output device; default is console terminal unless ofile is specified or the "=" is used, in which case the device is default; device is assumed to be online and write-enabled

ofile - name of file for directory (devo must be a file-structured device); default is DIR.TXT; if a file already exists with the specified name, it is autodeleted

/Q - do not rewind output medium prior to beginning directory search for file with same name as ofile (for tape units only); rewind if switch not specified

devi - device from which to take directory; system device is default; device is assumed to be online and ready

ifile - files to be listed in directory; wildcards are legal; default is .

/Q   do not rewind input medium prior to starting directory operation (tape devices only); rewind if switch not specified

/F - give short form of directory; long form if switch not specified

/B - list number of free blocks left on input medium (random access devices only)

/L - send directory to a line printer (parallel printers only)

There are samples of both forms of the directory on page 16 of this manual. Examples of the directory command:

```
    DIR DY0:DISK.TXT=MM0:/Q - The directory of files on MM0
```

N7

is written into a file called "DISK.TXT" on DY0. The tape is not rewound during the operation.

DIR =DR1:.BIN	- A directory of all files with "BIN" extensions on DR1 is written into a file called "DIR.TXT" on the system device.
DIR	- A directory of all files on the system device is typed at the console terminal.
DIR =	- A directory of all files on the system device is put into a file called "DIR.TXT" on the system device. Please take note of the effect of the equals sign on the operation of the directory command.
DIR DY0:/F/L	- A short form directory of all files on DY0 is printed on the line printer.

4.2.8 DO Command

The do command is used to execute an indirect command file for UPDAT. This file is a text file that contains one or more commands executable by UPDAT with the exception of EXIT. You can create a command file that accomplishes some common task such as building new media. This saves time and effort since you need not enter each command by hand.

The format of tne command is:

```
        DO file.ext
```

The specified file must be on the system device, therefore you cannot specify a device.

There are two functions available in the indirect command file in addition to the normal set of UPDAT commands. First, any command line beginning with a semicolon (;) will be treated as a comment. That is, no action will be taken; the line is merely printed. Second, a command line beginning with a dollar sign ($) will also be treated as a comment, except the processing of the command file will cease after the line is printed and resumed when a "Control X' 's typed. (Control X is typed by depressing the X key wh le holding the CTRL down.) This second function can be used to stop activity while the operator performs some requ red task such as mounting a new medium or placing a dev ce on ine.

The file can be made more global in scope by using logical un t numbers instead of device names in the commands. The user can then assign logical unit numbers prior to using the nd rect command file using the assign command. The example below illustrates the combined use of the two commands.

```
          Sample Command File:  RMBLD.TXT
          ------ ------- -----  ---------
            ZERO 1:
            CREATE DR1:
            FILE 1:=0:*.SYS
            FILE 1:=0:UPDAT.BIN
```

The above file can be used to build the XXDP System on any RM02/3 using any other XXDP medium. (Note that the command line conta ning a 'Y' only is required to verify the zero process. See the zero command description.) The process for do ng th s s:

C8

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

The underlined portion of the above example is that typed by UPDAT. The remainder is typed by the user.

4.2.9 DRIVER Command

The driver command is used to explicitly load a read/write device driver into memory. Up to two drivers may be loaded. If a third driver is loaded, one of the drivers currently in memory will be lost. If a requested driver is already in memory, no action is taken.

The format of the command is:

```
        DRIVER driver[/driver]
```

Driver is the two character device name (e.g.; DY = RX02).

The list of supported devices and their names is in Appendix C. Note that two devices may be specified with one command.

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

The purpose of the driver command is to allow a user to build XXDP media with limited resources. If the system device is required for building a new medium, the user can load the drivers required, remove the system medium, mount the new medium and build XXDP.

4.2.10 DUMP Command

The dump command is used to write the program image in the program buffer into a file on a medium. The image size is determ ned by the upper and lower memory limits displayed by the HICORE and LOCORE commands. A transfer address will be put into the file. This address can be examined and altered with the XFR command.

The format of this command is:

```
        DUMP [dev:]ofile[/Q]
```

dev - device to which file is to written; default is system device; device is assumed to online and write-enabled

ofile - file name for binary file; wildcards are not accepted; file with specified name must not already exist on device

/Q - inhibit rewind before searching for logical end-of-tape (tapes only)

Examples of the dump command:

```
        DUMP DY0:ZRLAA1.BIN
```

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

```
        DUMP FILE3.BIC
```

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

4.2.11 EOT Command

The EOT command is used to place a logical end-of-tape marker on a tape at the current position. Note, the tape is not rewound. All files after the current position will no longer be accessible. The marker consists of two consecutive tape marks. Any data beyond this point on the tape is lost.

The format of the command is:

```
        EOT dev:
```

The device must be a tape unit. The system device, if a tape unit, is the default device.

4.2.12 EXIT Command

The exit command is used to return control to the runtime monitor.

The format of the command is:

```
        EXIT
```

4.2.13 HICORE Command

The HICORE command is used to alter the address of the highest virtual memory location that is transferred during a dump' operation. The default location is printed after the LOAD command.

The format of this command is:

```
        HICORE
```

The address of the highest virtual memory location to be used during a dump operation is printed. UPDAT waits for you to alter or accept the location. You can enter a new location by typing the octal address or accept the given location by typing a "carriage return". This address must be above that of the lowest virtual location (low core) and below that of the top of the program buffer.
Examples:

```
        HICORE
        40000 45000
```

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

```
        HICORE
        100000 (CR)
```

This command allows the upper location to remain as 100000.

4.2.14 Load Command

The load command is used to load a binary file into memory for the purpose of modifying the program image. As the file is loaded, a checksum is computed and compared with a checksum stored with the file.

The command format is:

```
        LOAD [devi:]ifile[/N][/Q]
```

dev - device from which to load the file; default is the system device; device is assumed to be online and ready

file - file to be loaded; no default accepted

/N - inhibit printing of upper and lower memory limits and file name found (if wildcard used)

/Q - inhibit rewind before searching for file (tapes only)

Wildcards are permitted in the file specification, but you should be careful when using them. The data in the program buffer is the result of the overlays of each file found. The program buffer is not cleared between loads and the unused locations for the last file loaded does not necessarily contain zero's for contents. The wildcard feature is really only useful for doing file sanity checks to verify that files are not corrupted.

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). These parameters can altered by the user as described in subsequent command descriptions. Example:

```
        LOAD DY0:PROG1.BIN

        XFR:  000001   CORE:  000000,020000
```

This command is also used to load an XXDP monitor image into the program buffer. This must be done as part of the media build process. The section on New Medium Creation earlier in this chapter and Appendix E on Building XXDP describe this process.

4.2.15 LOCORE Command

The LOCORE command is used to alter the address of the lowest virtual memory location that is transferred in a dump operation. The default location is printed after the load operation is completed.

The format of this command is:

```
        LOCORE
```

Examples:

```
        LOCORE
        000200   0
```

This command modifies the low memory address to zero.

```
        LOCORE
        000000   20
```

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

4.2.16 Modify Command

The modify command is used 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
```

nnnnnn is the octal address of the virtual memory location whose contents are to be modified.

UPDAT responds by typing the address and the current contents. It then waits for you to either type new contents or accept the current contents by typing a "carriage return". If you wish to modify, or examine, two or more consecutive virtual memory locations, you type a "line feed" after modifying each location.

Examples: The following command modifies virtual memory location 2460:

```
        MOD 2460
        002460 770  771(CR)
```

The following command examines but does not modify virtual location 12004:

```
        MOD 12004
        012004 012736(CR)
```

The following command modifies two consecutive virtual memory locations (1220 and 1222).

```
        MOD 1220
        001220 120  167(LF)
        001222 120  1234(CR)
```

4.2.17 PIP and FILE Commands

The PIP and FILE commands are used to transfer a single file, or multiple files, from one medium to another. There are two differences between these commands:

1. The FILE command allows autodeletion; the PIP command does not.

2. The PIP command lets you specify an output file name; the FILE command does not.

Autodeletion is simply the removal of a file from the output medium if it has the same name and extension as the file being transferred. If you attempt to transfer a file when the name already exists on the output medium using the PIP command, the file is not deleted, the transfer does not occur and a warning message is printed. If several files are being transferred, only those that do not have names and extensions that match already existing files on the output medium are transferred. If a FILE command is used, all files are transferred regardless of what files exist on the output medium and any files on the output medium that have names and extensions matching those of files being transferred are deleted prior to the transfer. The operator is not notified of autodeletions.

The PIP command may be used to rename a file during the transfer process since output file specifications are allowed. The FILE command never accepts output file specifications and files will retain their names and extensions as they are transferred.

The format of these commands are:

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

devo - output device; system device is default; device is assumed to be online and write-enabled

ofile - file name for output file; wildcards are permitted, in which case the input files will be renamed to match the output specification (please refer to the examples below); default is .; if file already exists on output device, transfer will not occur; NOT USED WITH FILE COMMAND

/Q - do not rewind output medium prior to directory search for already existing file (tape devices only); rewind after each file if switch not specified

dev: - input device; default is system device; device is assumed to be online and ready

file - input file name; wildcards are permitted; default is .; file(s) specified must exist

/Q do not rewind before directory search for first file (tape devices only); rewind if switch is not specified

/N - do not type name of each file as it is found; type each name if switch is not specified

Examples of the PIP and FILE commands:

```
PIP DY0:NEW.BIN=DR1:ZZZZZZ.BIN
```

This command copies the file "ZZZZZZ.BIN" from DR1 to "NEW.BIN" on DY0. The transfer does not occur if "NEW.BIN" already exists on DY0.

```
FILE DY0:=DR1:ZZZZZZ.BIN
```

The file "ZZZZZZ.BIN is transferred from DR1 to DY0. It replaces a file of the same name on DY0 if such a file exists.

```
PIP =DD0:XMONCO.LIB
```

The file "XMONCO.LIB" is transferred to the system device from DD0. The name is not changed. If "XMONCO.LIB" already exists on the system device, you are given an error indication and the transfer does not occur.

```
FILE =DD0:XMONCO.LIB
```

This command has the same effect as the command in the previous example except that if "XMONCO" already exists on the system device, it is deleted prior to the transfer.

```
PIP DU1:=DU0:
```

All files on DU0 are transferred to DU1. Any files that already exist on DU1 are not transferred and you are notified. The remaining files are transferred.

```
        FILE DU1:=DU0:
```

In this case all files on DU0 will be copied to DU1, regardless of what files already exist on DU1. This command provides a convenient method of putting updated files onto an existing XXDP medium.

```
        PIP MM0:FILE??.*=
```

All files on the system device will be transferred to MM0 and RENAMED to have the characters "FILE" replace the first four characters of the original name. Be careful if you use wildcards on the output specification!

4.2.18 PRINT Command

The print command is used to read textual information from a file and output it to a line printer. The file must contain text in ASCII format.

The format of the command is:

```
        PRINT [dev:]ifile[/Q]
```

```
  dev - device where file is located; default is system
          device; device must be online

  ifile - file to be printed; must exist and contain text

  /Q - do not rewind before searching for specified file
          (tape devices only); rewind will occur if switch
          is not specified
```

Example:

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

This command reads the file 'HELP.TXT" from the tape on MM1 and prints it on the line printer. The search for the file begins at the current tape location; no rewind occurs.

4.2.19 READ Command

The read command is used to check device and media integrity. Each block of the file specified in the command is read into memory and a checksum is calculated. The computed checksum is compared to the checksum stored with the file.

The format of the command is:

```
        READ [dev:]file[/N][/Q]
```

dev - device from which file(s) are to be read;default is system device; device must be online

file - file(s) to be read; wildcards are accepted

/N - do not print name of each file as it is read

/Q - do not rewind before searching for specified files (tape devices only); rewind will occur if switch is not specified

4.2.20 RENAME Command

The rename command is used to change the file specification of an existing file without doing a transfer. The name of the file as recorded in the directory is changed, but there is no movement of data.

The format of the command is:

```
        REN [dev:]newnam=[dev:]oldnam
```

dev - device where file to be renamed exists; default is system device; device is assumed to be online and write-enabled; device must be same for both input and output

newnam - new file specification; file with same specification must not exist on device; wildcards are accepted

oldnam - current file specification; file must exist on device; wildcards are accepted

Examples of the rename command:

```
        REN DX1:DIAG.OLD=DX1:DIAG.BIN
```

This command renames to the file DIAG.OLD on DX1 to DIAG.BIN.

4.2.21 TYPE Command

The type command is used to read textual information from a file and output it to the console terminal. The file must contain text in ASCII format.

The format of the command is:

```
        TYPE [dev:]ifile[/Q]
```

```
  dev   device where file is located; default is system
          device; device must be online

  ifile - file to be typed; must exist and contain text

  /Q - do not rewind before searching for specified file
          (tape devices only); rewind will occur if switch
          is not specified
```

Example:

```
        TYPE SYSTEM.CCC
```

The file "SYSTEM.CCC" is be read from the system device and printed on the console terminal.

4.2.22 XFR Command

The XFR command is used to modify the transfer address in the program that has been loaded. The file created by the DUMP command will have this transfer address. This is address of the location to which control will be transferred when the program is started.

After you enter this command, UPDAT prints the current transfer address. If you don't wish to alter the transfer address, you should immediately type a "carriage return".

The format of the command is:

```
        XFR
```

Examples of this command: The following command changes the transfer address from 200 (octal) to 1000 (octal):

```
        XFR
        000200 001000
```

The following command examines, but does not change, the transfer address:

```
        XFR
        002000(CR)
```

4.2.23 ZERO Command

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

CAUTION:

all data on the medium prior to a zero operation is irretrievably lost after the operation. UPDAT makes no attempt to determine what is on a medium and will destroy customer data.

A warning is printed whenever this command is invoked, stat ng which device is involved. You must then verify that the zero operation is to take place.

The format of the zero command is:

```
ZERO dev:
```

There is no default for the device. The device must be online and write-enabled. The following warning will be issued after the command is entered:

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

The only answer that will confirm the user's intent to carry on is "Y".

There will be an additional warning message if you specify the system device in the ZERO command.

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

If you wish to proceed with the process, you must type a "Y". The meaning of the warning is that you must assure the presence in memory of two drivers, one for the system device and one for the device from which files will be moved to the new med a in the system device. To assure that the necessary dr vers are in memory, use the DRIVER command.

4.3 Sample Modification

The following sample file modification illustrates the use of UPDAT commands. It is based on the following patch for a fictitious 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. The underlined portion is that typed by UPDAT. (LF) and (CR) refer to line feed and return.

```
.R UPDAT

CHUP2B0 XXDP UPDAT UTILITY

RESTART ADDRESS:  002432

LOAD ZXXXB0.BIN

XFR:  000001 CORE:  000200, 02723

MOD 1224
001224 106701 240(LF)
001226 177660 240(CR)

MOD 1224
001224 000240 (LF)
001226 000240 (CR)

MOD 1452
001452 000376 374(CR)

MOD 1452
001452 000374 (CR)

DUMP ZXXXB1.BIN
```

The user examines locations previously modified in order to verify the changes. The user also renames the file to reflect the patch level.

CHAPTER 5

PATCH UTILITY

The PATCH program can be used to modify any binary formatted (.BIN or .BIC) file stored on an XXDP storage medium. It is an alternative to the LOAD-MOD-DUMP sequence of UPDAT.

5.1 Description

PATCH should be used in the following instances:

1. You are modifying a file that is too large to be loaded into the memory space of your system. This situation precludes the use of the LOAD-MOD-DUMP sequence of the update programs.

2. You are modifying DEC/X11 runtime exercisers. (It is assumed that reader is familiar with DEC/X11 usage. If not, please read the DEC/X11 User Manual before attempting to use PATCH on this software.) As these programs CANNOT be patched using the update programs, you must use this program if you wish to produce a permanently modified .BIN file for a DEC/X11 RTE.

```
**********************************
        Notice
The DEC/X11 features have
not been fully implemented in
DEC/X11 monitor.  Please
patch RTE's with this
as you would any other
binary file.
**********************************
```

5.1.1 Starting PATCH

Operation of this program consists of two phases. The first 's the building of a table containing the modifications that will be made to the file in question. This table is referred to as the "input table". You fill this table with the addresses you wish to modify within the file, along with the desired contents of these addresses. This table may then be saved as a file and retrieved for later use. The second phase of operation is the combining of the 'nformation contained in the input table with the actual binary f'le to produce a new, modified file. The original file 's not modified by the program.

In order to load and start this program, you must type the following command to the XXDP monitor:

```
        .R PATCH
```

Th's command causes the PATCH utility to be loaded into memory and begin executing. The program identifies itself w'th the message:

```
        CHUPA?? XXDP PATCH UTILITY
```

The program then prompts for your input with the "*" character. At this point you can begin entering commands to the program.

5.1.2 Command Summary

The valid commands for PATCH are:

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	Pr'nt input table on terminal

In order to patch a file with this program you must perform two operations:

1. You must build an input table containing all of the addresses which you wish to modify within the file, along with the contents you want these addresses to have. The input table may have a maximum of fifty (50) of these entries.

   The commands you may use to build the input table are:

   CLEAR
   GETM
   MOD
   TYPE
   KILL
   SAVP
   GETP

   These commands are described below.

2. After you have completed the input table you must use the PATCH command to add the address modifications within the input table to the file you want to patch. The PATCH command is described below.

It is important to note that the file you are modifying is never completely loaded into memory.

5.2 PATCH Commands

5.2.1 BOOT

The BOOT command boots the specified device.

The format is:

```
        BOOT [dev:]<CR>
```

The default device is the system device.

5.2.2 CLEAR

The CLEAR command clears the input table of all entries.

The command format is:

```
        CLEAR
```

5.2.3 EXIT

The EXIT command returns control to the XXDP monitor.

The command format is:

```
        EXIT
```

5.2.4 GETM

This command is used only when patching DEC/X11 runtime exercisers. It retrieves a DEC/X11 "MAP" file of the specified filename from the specified device and loads it into memory (see section: "DEC/X11 MAP FILES").

The command format is:

```
        GETM [dev:]filnam.ext
```

The default dev'ce is the system device.

5.2.5 GETP

The GETP command causes the input table to be loaded with the contents of a file that was created using the "SAVP" command. Execution of this command causes any previous contents of the input table to be lost.

The command format is:

```
        GETP [dev:]filnam.ext<CR>
```

The default device is the system device.

5.2.6 KILL

The KILL command is used to delete an entry from the input table.

The command format is:

```
        KILL <addr>
```

5.2.7 MOD

The MOD command is used to enter an address and the desired new contents of that address into the input table. The MOD command has two modes of operation, depending on whether you are modifying a DEC/X11 RTE or another type of binary file. These two modes are described in the following sections.

5.2.7.1 Binary (NON DEC/X11) Mode

When used with binary files other than DEC/X11 RTE's, the format of the MOD command is:

```
        MOD addr
```

where addr is any valid 16-bit address. Leading zeros can be omitted.
After you press carriage return to conclude the command, the requested address is retyped, followed by a slash. If this address has not been previously entered in the input table, the slash is followed by six dashes. (Because the file you wish to modify is not in memory, there is no way of knowing the current contents of the location you have specified.) If the address has been inserted into the input table, the previously entered contents of the address are typed after the slash.

Example:

```
        MOD 123456
        123456/------
```

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

Example:

```
        MOD 11040
        011040/000240
```

In this case an actual value appears after the slash. This indicates that the operator had previously entered this address into the input table and had specified the new contents of the address to be 000240.

At this point you can type the value you wish to have loaded into this address. This value can be any octal number from 0 to 177777. Leading zero's may be omitted. After entering the value, you can type either a <CR> (carriage return) or a <LF> (line feed). A <CR> closes the table entry for this address and causes a prompt to be printed so that another command can be typed. A <LF> closes the current table entry and makes a new entry for the next addressable memory location (i.e. <addr>+2). The new contents for this address can then be typed.

Example:

```
        MOD 123456<CR>
        123456/------ 000207<CR>
```

In the example, the operator has specified that location 123456 should contain 000207. The carriage return closes the input table entry and causes a prompt () to be printed.

Example:

```
        MOD 11040<CR>
        011040/000240 000137<LF>
        011042/------ 051502<CR>
```

In this case the operator has re-opened the table entry for address 11040, changed the contents to 000137, and typed a <LF> to make a table entry for location 11042. The dashes indicate this location had not been previously entered into the table. This location receives a contents of 51502, then a <CR> is typed to close the table entry and cause a prompt to be printed.

5.2.7.2 DEC/X11 Mode

When working with DEC/X11 files, the command has three different formats, as follows:

```
        MOD addr
        MOD MON modnam addr
        MOD opmod addr
```

Form (1) of the MOD command is the same for DEC/X11 usage as it is for non-DEC/X11 usage (see previous section), with the exception that 18-bit addresses accepted.

Forms (2) and (3) of the MOD command allow the operator to specifiy locations within a DEC/X11 runtime exerciser (RTE) by typing the name of a monitor module or option module followed by an offset into that module. These forms of the command may be used only if a DEC/X11 "map" file has been retrieved by means of the GETM command (see section: 'DEC/X11 MAP FILES").

Format (2) is used for modifying locations within monitor modules. An example of this format is:

```
        MOD MON KTERR 24<CR>
```

The keyword MON is used to indicate that the module is in the monitor section of the RTE. In this case the operator is specifying location 24 relative to the beginning of the monitor module named "KTERR".

Format (3) is used for modifying locations within option modules of exercisers. Opmod is the name of the option module to be modified. The name has five characters, the fifth character being the copy number (the first copy is 0). An example of this format is:

```
        MOD CPBJ0 100<CR>
```

The operator, by typing this command, is indicating that he wishes to modify location 100 relative to the beginning of the first copy of option module CPBJ.

In all three formats, after the <CR> is typed the actual physical address being referenced will be typed, followed by a slash (/). If this address has not been previously entered in the input the slash will be followed by six dashes (------). If the address has already been entered in the input table, the previously entered contents of the address will be printed after the slash.

Example:

```
        MOD 012546<CR>
        12546/------
```

In the example, the operator has specified physical address 12546 to be modified. The address is retyped, followed by a slash. The dashes indicate that this is the first time this address has been specified for modification.

Example:

H10

```
        MOD MON KTERR 10 <CR>
        007126/000240
```

In this case the operator has specified an address within the monitor section of the RTE and has done so by using a module name. Notice that the program determines the actual physical address represented by the command string arguments and prints that address. Here an actual value appears after the slash. This indicates that the operator had previously entered this address into the input table and specified new contents of 000240.

At this point you can type the value you wish to have loaded into the specified address. This value can be any octal number from 0 to 177777. Leading zeros can be omitted. After entering the value you wish, you can type either a carriage return or a line feed. A <CR> closes the current table entry and causes a prompt () to be printed so that you can then type another command. A <LF> closes the current table entry and makes a new entry for the next addressable memory word (i.e., <addr>+2>. You can then type new contents for the address.

Example:

```
        MOD 012546<CR>
        12546/------ 000207<CR>
```

In the example, the operator has specified that location 12546 should contain 000207. The <CR> closes the input table entry and causes a prompt to be printed.

Example:

```
        MOD MON KTERR 10<CR>
        007126/000240 137<LF>
        007130/------ 51502<CR>
```

In this case the operator modified the contents of the location 7126 so it will be 000137 (he had previously specified that this address should contain 000240). He then typed a <LF> to close the table entry for address 7126 and to create a table entry for location 7130. The dashes indicate that this location had not been previously entered in the table. This location received a contents of 51502 and a <CR> was typed to close the table entry and cause of prompt to be typed.

5.2.8 PATCH

After the device address modifications have been entered in the input table, a new output file containing these modifications can be produced with the use of the PATCH command. This command reads the specified input file, adds the address modifications contained in the input table, and builds a new output file having the specified file name.

The format of the command is:

```
        PATCH [dev:]filnam.ext=[dev:]filnam.ext<CR>
```

The default device (input and output) is the system device. The input file and the input table are unaffected by the execution of this command.

An example command string is:

```
        PATCH DY1:SAMPL2.BIN=DY0:SAMPL1.BIN<CR>
```

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.

After you press the carriage return to conclude the command, the following instruction is printed:

```
IF THIS IS DECX11 TYPE THE MONITOR TYPE.  ELSE JUST <CR>
```

If you are patching any file which is not a DEC/X11 runtime exerciser, then type a <CR>. If this response is typed, the program begins construction of the output file. When execution is completed, the message "DONE" is printed, followed by a prompt.

If you are patching a DEC/X11 runtime exerciser (RTE), you must respond to the printed question by telling the program the type of monitor contained in the RTE. There are three methods for determining the monitor type of the RTE:

1. Run the DEC/X11 configurator/linker program and type the configuration file for this RTE, if it exists.

2. Run the DEC/X11 configurator/linker program and type the MAP file for this RTE, if it exists. The monitor type appears at the top of the listing.

3. Run the RTE. The monitor type is printed at start-up time.

If the monitor of the RTE is one that does not support memory management, the program now begins building the new output file. When this process is complete, the message "DONE" is printed on the terminal, followed by a prompt. If, on the other hand, the specified DEC/X11 monitor type is one that does support memory management, the program now checks to see if a MAP file has been loaded into memory with the GETM command. If there is no MAP file in memory, the following instruction is typed:

```
        TYPE MODQ ADDRESS:
```

In order to determine this address, you must look at a listing of an appropriate MAP file (see DEC/X11 MAP Files). The symbol MODQ is located within the monitor module "CONFIG", so just find the module name "CONFIG" on the listing, then look at the symbol names underneath the module name until "MODQ" is found. The physical address printed next to the symbol "MODQ" is the address which must be typed in response to the question. If a MAP file was previously loaded into memory using the GETM command, the program automatically finds this address and thus does not ask the question.

Next, the following message is printed:

```
        IF MODIFYING OPTION MODULES, TYPE LOWEST MODULE
          ADDRESS, ELSE JUST <CR>
```

If you are modifying only the monitor section of the runtime exerciser, the proper response to this message is to simply type a <CR>. If you are patching both the monitor area and one or more option modules, or if you are patching only option modules, you must now type the address of the first option module that was linked into the runtime exerciser. This address can be obtained in two ways:

1. If you have a MAP file listing for the proper monitor type (see "DEC/X11 MAP FILES"), find the first occurance of an option module name on the listing and use the physical address associated with that option module. The option modules 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".

2. Before running the PATCH program, you can load and run the RTE that you intend to patch, then type a "MAP" command. This command causes the starting addresses of all option modules to be printed. For each option module, look for the address labelled "PA:". Find the physical address which is lowest (not necessarily the first one printed!).

After the typing of a <CR> the program will commence construction o *he output file. When execution has been completed the m sage "DONE' will be printed, followed by a prompt.

5.2.9 SAVP

This command causes the contents of the input table to be saved as a file with the specified filename on the specified device. The command does not cause any alteration to the input table contents.

The command format is:

```
        SAVP [dev:]filnam.ext<CR>
```

The default device is the system device.

5.2.10 TYPE

The TYPE command causes the contents of the input table to be listed on the system terminal.

The command format is:

```
        TYPE
```

5.3 DEC/X11 MAP Files

In order to utilize the full capabilities of the MOD command when modifying DEC/X11 runtime exercisers, you must have a MAP file.

The MAP file is produced by the DEC/X11 configurator/linker. It is the symbol table which is generated at link time and saved using the SAVM command. (Please refer to the DEC/X11 User Manual.) Without the MAP file the MOD command will only accept physical addresses as arguments, but if a MAP file has been fetched using the GETM command, the MOD command will accept module names (both monitor and option module names) as arguments.

If you are going to modify the option modules of a particular runtime exerciser, and if you wish to be able to type the option module name and an offset value when using the command, then you must use the MAP file generated during the linking of that particular RTE because the number and order of the option modules in any RTE is unique. If you don't have the proper MAP file you must manually calculate the physical address of the option module's relative address and type that value as the MOD command's argument.

On the other hand, for any given monitor type (A, B, C, etc.), the monitor modules are always linked in the same sequence. This implies that if you are modifying the monitor section of an RTE and wish to type the monitor module name plus an offset when using the MOD command, you need not have the MAP file for the particular RTE you are modifying. Any MAP file for the proper monitor type will do. For example, if you are modifying the monitor area of an RTE of a monitor type C, you may use any MAP file which was generated when linking any RTE having monitor type C. Similarly, the address of MODQ is also the same for every monitor of the same type, so if the PATCH command prompts you for the address of MODQ, you may look at any map listing of the proper monitor type to obtain the address.

It is important to remember that when a new release of the DEC/X11 monitor library is issued, all MAP files generated from the previous release of the library become invalid. They may not be used when patching files generated with the new library. New MAP files must be produced.

5.3.1 Suggested DEC/X11 Application

When a 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 are in an environment in which you build many RTE's (manufacturing, for example), it is suggested that you build and save an input table for every monitor type. These saved tables can be added to and re-saved every time a new DEPO is issued. After any RTE is built with the configurator/linker, simply run this program, get the input table for the proper monitor type, add to the table any modifications that must be made to the option modules (using the MAP file if you wish) and then execute the PATCH command.

CHAPTER 6

SETUP UTILITY

SETUP 's an XXDP utility that elts you build the hardware and software tables for a diagnostic prior to running the diagnost'c and store the tables with the diagnostic. SETUP has the same memory requirements as DRS: 5.75K words. The m'n'mum size system that can be used is 28K words.

6.1 Starting SETUP

To run SETUP, use the XXDP run command. Below is an example of start'ng SETUP for use 'n building XXDP environment diagnostics. The underlined portions are typed by the system.

```
        .R SETUP
        *
```

SETUP is now ready to accept commands.

6.2 Commands

The following are the SETUP commands:

SETUP	build tables for specified diagnostic
LIST	type a list of DRS diagnostics on a medium
EXIT	return control to XXDP

6.2.1 LIST

The LIST command is used to obtain a list of all DRS compatible diagnostics on a medium.

The format of the command is:

```
        LIST [dev:][file.ext]


  dev - device to search for DRS-compatible files; default
        is the system device

  file.ext - file(s) to search; extension must be BIN or
        BIC; wildcard specifications are accepted; default
        is "*.BI?"
```

6.2.2 SETUP

The SETUP command causes the specified diagnostic to be loaded into memory. SETUP then processes the table building code in the diagnostic. You go through the same process that occurs when actually running the diagnostic and issuing a START command.

The format of the command is:

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

  devo - device to which file is to be written; default is
        system device; device must be on-line

  ofile - name of file for the diagnostic that has been
        SETUP WITH .BIN OR .BIC EXTENSION

  devi - device from which file is to be read; default is
        system device; device must be on-line

  ifile - name of file for the diagnostic that is to be
        SETUP
```

You can give the output file the same name as the input file, but you get a message that warns against the accidental loss of the original file. The message is:

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

If you type a "Y" or no answer at all, the input file

file is deleted after the SETUP process and the new file
is then written.

CHAPTER 7

XTECO UTILITY

The XTECO (pronounced 'ex-tee-co") utility is used to create and edit (modify) text files. Text files contain ASCII data representing valid text. Valid text consists of all printing characters, tab, carriage return, line feed and form feed. The principal text files in XXDP are batch control (chain) files. XTECO is a simple editor, a limited subset of the TECO character editor supported by most of DEC's operating systems. The commands are few and simple, but adequate for the task.

7.1 Description

Before describing the commands in detail, let's look at some basic XTECO concepts. To XTECO, a text file contains one long string of characters sort of like beads on a string. It processes the file one character at a time. The utility can only have a certain number characters in memory, so only a segment of the string can be worked with any given time. There are special characters (carriage return and line feed) that act as signals to XTECO to tell it when one line of text ends and another begins. This allows the editor to manipulate lines of text as well as characters. The editor keeps track of where it is on the string with a pointer. This pointer is manipulated by the various commands and is used by XTECO to locate where new text is to be placed and old text removed or modified. This pointer may be moved back and forth over the portion of the string in memory at the time, but cannot go backwards into the portion of the string already processed and placed into the new file. Thus you need to understand two basic concepts: text as a string of characters and a pointer to locate the editor on that string.

As previously mentioned, only a certain segment of a text file may reside in memory at any given time. XTECO processes files by reading a segment from the input file into memory, doing any edit functions on that segment (as directed by the user) and then writing the segment into the output file. This process continues until the entire input file has been acted upon. In the case of the "TECO" command (see next section) where the input and output file may have the same name and reside on the same medium, XTECO creates a temporary output file which replaces the input file after the edit process is complete.

7.1.1 Starting the Edit Process

The first thing a user must do after starting XTECO is to initiate the process of editing text. (XTECO is started by typing "R XTECO" in response to the XXDP monitor prompt.) There are three commands to do this.

```
     TEXT - used to create a new file
     TECO - used to modify a file
     EDIT - used to modify a file
```

Each of these commands puts XTECO into what is called "edit mode". While in this mode, text information can be created, deleted or modified. There is one difference between the EDIT and TECO commands. The TECO command can be used with random-access devices (i.e., disks) only and the input file name is all that is required. The output file name ias the same as that of the input file. An output file is created during the edit process and is used to replace the input file after completion of the edit process. The EDIT command can be used with any type of device, but the input and output devices must be different (either different device types or different units of the same type).

The formats of these commands are:

```
          TEXT [devo:]ofile
          TECO [devi:]ifile
          EDIT [devo:]ofile=[devi:]ifile
```

where,

```
   devo - output device where the new file is to be stored
   ofile - name of file for output
   devi - input device from which old file is to read
   ifile - name of input file
```

After one of these three commands is executed, XTECO is

in 'edit mode" and issues a double quotation mark ('"') as a command prompt instead of an asterisk.

7.1.2 Command Summary

The thirteen commands available in edit mode are listed by type of function in the following table:

Edit Mode Commands

```
    Pointer Location
      L - move the pointer line by line
      C - move the pointer character by character
      J - move the pointer to the beginning of text
      ZJ - move the pointer to the end of text

    Search
      S - search for specified string in text now in memory
      N - search for specified string in remainder of text file

    Modify/Display Text
      T - type text
      D - delete character(s)
      K - delete line(s)
      I - insert text
      A - append text to that currently in memory

    Terminating Edit Mode
      EX - exit edit mode
```

All commands are terminated by two **altmode (escape)** characters. The altmode (escape) key is usually the left-uppermost key on DEC terminals. The altmode (escape) character is echoed on the terminal as "$" and is represented that way in examples in this chapter.

7.2 Commands

The following sections explain each edit mode command in detail. Following these explanations, there is a sample edit session.

In examples where it is important to display the position of the pointer, a caret, "↑", is used to designate the position within the sample text. This is for illustrative purposes only! This character is not used for this purpose in actual operation.

7.2.1 A Command

The A command is used to increase the amount of text stored 'n memory. This is done by reading the next section of text from the input file and "appending" it to the text already in core. You may append as many sections of text as memory s'ze limits allow.

The format of the command is:

```
        A$$
```

The combined sections of text in memory are treated as a s'ngle section by the previously described commands.

7.2.2 C Command

The C command is used to move the pointer on a character-by-character basis. Lines are not recognized by this command; the carriage return/line feed sequence is treated as two characters. The pointer may be moved forward or backward any number of characters with'n the text currently stored in memory.

The format of the command is:

```
        [n]C$$
```

where n is an optional argument that specifies the direction to move and the number of characters encompassed by the move. It is a decimal number that is positive for forward motion and negative for backward motion. If it is not specified, "1" is assumed.

Following are examples of the C command. The caret, "↑" indicates the position of the pointer after execution of each command. XTECO's prompt ('"') is also shown in the examples.

```
     Command        Text in Memory

      --          ;NEXT COMMAND WILL T↑EST RP06
                  ;ALL ERRORS WILL BE REPORTED

     "C$$         ;NEXT COMMAND WILL TE↑ST RP06
                  ;ALL ERRORS WIL 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
```

Please note the effect of the carriage return/line feed sequence on the execution of the last command in the example above. The sequence counts as TWO characters.

7.2.3 D Command

The D command is used to delete characters from the text in memory. Any number of characters, either preceeding or following the current pointer position, may be deleted.

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 pointer position and is negative if they preceed it. If it is not specified, "1" is assumed.

Following are examples of the D command. The caret, "↑", indicates the position of the pointer. XTECO's prompt ('"') is also shown in the examples.

```
     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

The EX command is used when all editting operations have been completea. The output file is closed. If the edit session was initiated using the TECO command, the input file is renamed with a .BAK extension and the output file will be given the original n_ e of the input file. XTECO will no longer be in edit mode as signified by the switching of the prompt character back to an asterisk ("*") from a double quote ('"').

The format of the command is:

```
        EX$$
```

7.2.5 I Command

The I command is used to insert new text. The text is inserted after the current pointer position. The pointer will be positioned after the new text upon completion of the insertion.

The format of the command is:

```
        Itext$$
```

where "text" is the text to be inserted.

The inserted text can consist of any valid text characters. Valid text characters are all printing characters, tab, carriage return, line feed and form feed.

Following are examples of the I command. The caret, "↑", indicates the position of the pointer after execution of each command. For purposes of illustration, line terminators are depicted in these examples. The return typed by the user is represented as "<RET>"". The two character sequence generated by the return and stored in the text is represented as "<CR><LF>". XTECO's prompt (''') is also shown in the examples.

```
        Command                 Text in Memory
         --                    R UPD2<CR><LF> ↑

        'IPIP$$                R UPD2<CR><LF>
                               PIP↑

        'I DK0:=DK1:<RET>      R UPD2<CR><LF>
        $$                     PIP DK0:=DK1:<CR><LF>
                               ↑

         COMMAND                     TEXT IN MEMORY

         --                    FILE DK1:=DK↑:<CR><LF>

        'I2$$                  FILE DK1:=DK2↑:<CR><LF>
```

7.2.6 J Command

The J command is used to position the pointer at the beginning of all text currently stored in memory.

The format of the command is:

```
        J$$
```

The command is terminated by two altmodes or escapes which are echoed on the terminal as '$$' as shown above.

Following is an example of the J command. The caret, '↑', indicates the position of the pointer before and after command execution.

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

The K command is used to delete lines of text from the text stored in memory. A line of text is a string of characters between carriage return/line feed sequences. (This sequence is produced by typing the "return" key on your terminal.) Deletion of a line includes the deletion of the terminating carriage return/line feed sequence in addition to the characters in the line. Any number of lines may be deleted either preceeding or following the current position of the pointer. If the pointer is positioned within a line, not all of the line will be deleted (see examples that follow).

The format of the command is:

```
        [n]K$$
```

where n is an optional argument that specifies the number of lines to be deleted. It is decimal number and is positive if the lines preceed the pointer and negative if they follow it. If this argument is not specified, "1" is assumed.

Following are examples of the K command. The caret, "↑", indicates the position of the pointer after the execution o each command. XTECO's prompt ('"') is also shown in the examples.

```
     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 pointer 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

The L command is used to move the pointer on a line-by-line basis. A line of text is a string of characters between carriage return/line feed sequences. (This sequence is produced by typing the 'return" key on your terminal.) The pointer can be moved either backward or forward any number of lines in the text currently stored in memory. The pointer is always positioned at the beginning of a line after execution of the L command.

The format of the command is:

```
        [n]L$$
```

where n is an optional argument that specifies the direction to move and the number of lines encompassed by the move. It is a decimal number that is positive for forward motion and negative for backward motion. If it is not specified, "1' is assumed.

Following are examples of the L command. In these examples, a caret, "↑", indicates the position of the pointer after execution of each command. XTECO's prompt (") is also shown in the examples.

```
     Command         Text in Memory

      --          R ZRLA↑??
                  IF ERROR THEN
                  PRINT RL01 HAS HARDWARE PROBLEM
                  END

     'L$$         R ZRLA??
                  ↑IF ERROR THEN
                  PRINT RL01 HAS HARDWARE PROBLEM
                  END

     '-1L$$       ↑R ZRLA??
                  IF ERROR THEN
                  PRINT RL01 HAS HARDWARE PROBLEM
                  END

     '2L$$        R ZRLA??
                  IF ERROR THEN
                  ↑PRINT RL01 HAS HARDWARE PROBLEM
                  END
```

7.2.9 N Command

The N command has the effect of a "non-stop" S command. It is exactly like the S command, except that it searches through all remaining text in a file.

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, carriage return and line feed.

The editor searches through text in memory, starting from the current pointer position, until either a match is found or the end of text is encountered. If the editor fails to find a match for the specified string within the text currently in memory, it writes the text in memory into the output file and brings more text in from the input file. This process continues until either a match is found or the entire input file has been checked.

The J command can not be used to recover from a failed search if that search caused the section of text that you started from to be written to the output file. In this case, you must exit edit mode and re-enter with the previous output file as input.

7.2.10 S Command

The S command causes the editor to search for a specified string of characters in the text currently stored in memory. Searches take place in the forward direction only. The search encompasses the text in memory only.

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, including tab, carriage return and line feed.

The editor searches through text in memory, starting from the current pointer position, until either a match is found or the end of text is encountered. An error message is printed if a match is not found. The pointer is positioned AFTER the string found by the search or after all text in memory if no match is made. (You can reposition the pointer to the beginning of text in memory after a failed search by using the J command.)

Following are examples of the S command. The caret, "↑", indicates the postion of the pointer after execution of each command. XTECO's prompt ('"') is also shown in the examples.

```
      Command        Text in Memory

       --           ↑R UPDAT
                    PIP DXO:=DX2:*.BIN
                    EXIT

      "SDX1$$       R UPDAT
                    PIP DXO:=DX2:*.BIN
                    EXIT↑

      "J$$          ↑R UPDAT
                    PIP DXO:=DX2:*.BIN
                    EXIT

      "SDX2$$       R UPDAT
                    PIP DXO:=DX2↑:*.BIN
                    EXIT
```

In the above example, the first search failed. The editor would print an error message:

```
NOT FOUND:  DX1
```

7.2.11 T Command

The T command is used to type text on the console terminal. The text typed 's relative to the position of the pointer in the text currently stored in memory. Typing is line-by-line and any number of lines before or after the current pointer position may be typed. A line of text is a string of characters between carriage return/line feed sequences. (This sequence is produced by typing the "return" key on your terminal.) If the pointer is positioned within a line of text, typing starts/concludes at the pointer position (see examples).

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 pointer and negative if the preceed it. If n is not specified, "1" is assumed.

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

Following are examples of the T command. The caret, "↑", in the sample text indicates the position of the pointer. XTECO's prompt ('"') is also shown in the examples.
First sample text:

```
      ;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
   '2T$$      R ZDZA??
              R ZDZB??
```

Second sample text:

```
      R PROG1
      R PRO↑G2
      R PROG3
```

```
   Command            Text Typed

    T$$              G2
  '-1T$$             R PROG1
                     R PRO
  'OT$$              R PRO
   HT$$              R PROG1
                     R PROG2
                     R PROG3
```

7.2.12 ZJ Command

The ZJ command is used to position the pointer after all text currently in memory.

The format of the command is:

```
         ZJ$$
```

Following is an example of the ZJ command. The caret, "↑", indicates the position of the pointer before and after command execution.

Initial state:

```
      ↑R PROG1
      R PROG2
      R PROG3
```

After execution of the ZJ command:

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

2 I Command The I command is used to insert new text. The text is inserted after the current pointer position. The pointer will be positioned after the new text upon completion of the insertion.

The format of the command is:

```
        Itext$$
```

where "text" is the text to be inserted. This text can consist of any valid text characters. Valid text characters are all printing characters, tab, carriage return, line feed and form feed.

Following are examples of the I command. The caret, "↑", indicates the position of the pointer after execution of each command. For purposes of illustration, line terminators are depicted in these examples. The return typed by the user is represented as "<RET>"". The two character sequence generated by the return and stored in the text is represented as "<CR><LF>". XTECO's prompt ('"') is also shown in the examples.

```
      Command                   Text in Memory
       --                     R UPDAT<CR><LF> ↑

      'IPIP$$                 R UPDAT<CR><LF>
                              PIP↑

      'I DK0:=DK1:<RET>       R UPDAT<CR><LF>
      $$                      PIP DK0:=DK1:<CR><LF>
                              ↑

              COMMAND                     TEXT IN MEMORY

       --                     FILE DK1:=DK↑:<CR><LF>

      'I2$$                   FILE DK1:=DK2↑:<CR><LF>
```

7.3 Combining Edit Commands

The user can combine several edit mode commands on a single command line. This is done by separating each command by a single escape (altmode) character and then terminating the entire string of commands by two escapes. For example, the following commands:

```
        *NTEST$$

        *0T$$

        *T$$
```

can be combined into a single string:

```
        *NTEST$0T$T$$
```

In both cases, XTECO will do a non-stop search for the string "TEST", type the characters from the beginning of the line where the string was found to the current pointer position and then type the characters from the current pointer position to the end of the line. Combining commands is merely a convenience for the user. It is suggested that you do not attempt to combine commands until you are familiar with the operation of individual commands. The next section has sample edit sessions that show both methods.

7.4 sample Edit Sessions

What follows is a series of sample edit sessions that show you the various ways of handling XTECO. In these examples, the underlined text is that which is typed by XTECO. At appropriate locations, comments have been included for reader assistance. These comments are enclosed in square brackets ([]) and should not be confused with the actual dialogue that is taking place between user and software.

7.4.1 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
.
```

7.4.2 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
```

7.4.3 Use of Combined Edit Commands

```
.R XTECO

EDIT DL1:TEST1.CCC=TEST.CCC

[EDIT TEST.CCC, WHICH IS ON THE SYSTEM

DEVICE, AND PLACE THE EDITTED 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

.
```

7.5 Non-edit Commands

There are three XTECO commands that are not related to actual text file editt ng. These commands are provided for user convenience.

7.5.1 TYPE and PRINT

The TYPE and PRINT commands are used to print text files on the console terminal and line printer respectively. They are equivalent to the UPDAT commands of the same name.

7.5.2 EXIT

The EXIT command is used to return control to the XXDP monitor.

CHAPTER 8

BATCH CONTROL

XXDP has a facility for running programs without operator intervention. This facility is called batch control or chaining. The commands that would normally be issued by an operator are put into a text file (using XTECO) and the monitor processes the commands in this file rather than requiring an operator to enter each command manually.

8.1 Description

Once a batch control file has been created, it can be used over and over again. The batch control process releases you from having to do repetitive tasks such as building new media or running a common set of diagnostics. More importantly, batch control lets you develop a test strategy and use the strategy consistently. This is done by selecting the proper diagnostics and running them in a particular order and mode to achieve the best test process. Once the process is developed, it is put into a batch control file.

Older versions of XXDP had very limited batch control services. Essentially you could 'chain' together a series of run commands which would run various diagnostic programs, such as:

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

You could intermix comments to be printed as the chain was processed. This primitive process was adequate for most simple procedures, but was not adequate for more sophisticated operations such as update kits.

8.1.1 Batch Control Functions

XXDP now contains a fairly sophisticated set of batch control functions listed below. These functions and techniques for using them to run diagnostics and utilities are described here.

Batch Control Functions

Monitor Commands	monitor commands: R, L, S, C, and E)
Utility Commands	UPDAT, SETUP, etc.
DRS Commands	all DRS commands and diagnostic dialogue

8.1.1.1 Monitor Commands

Certain of the monitor commands described in Chapter 2 can be used in a batch control file. They are the R, L, S, C and E commands. There are two functions which are different when used under batch control instead of operator control. First, the R (Run) Command has a pass switch for use with diagnostic programs which are not DRS compatible. The diagnostic may be run a certain number of passes by using the switch as shown:

```
        R DIAG/5
```

DIAG will run 5 passes before the batch operation continues on.

The C (Chain Command) may be used in a batch file with one restriction. Batch operations can be nested one level only. That is, a batch file may start another batch file and then continue after the second file has been processed, but the second batch file may not start another (third) file.

With these exceptions, the monitor commands function under batch control as they would under operator control.

8.1.1.2 Utility Commands

The commands for various XXDP utilities may be used in batch control operations.

8.1.1.3 DRS Commands

All of the DRS commands described in Chapter 3, including all switches and flags, can be used in a batch control file. All dialogue that would normally take place between an operator and a DRS diagnostic can also be placed in a batch control file. Batch Commands

8.1.1.4 Batch Control Command Summary

The following batch control commands can be used:

conditionals	sections of the batch file can be processed conditionally under operator control or runtime conditions
GOTO tag	begin processing at another section of the batch file designated by "tag"
PRINT	temporary override of QUIET
QUIET	inhibit printing of batch file if printing or enable printing if printing was inhibited previously
QUIT	terminate the batch operation
SMI/CMI	enable/disable manual intervention operations in specialized diagnostics
WAIT	stop batch operation until the operator types a Control X

8.2 Batch Control Commands

8.2.1 Conditionals

Sections of a batch control file can be processed, or not processed, based on either operator input or certain conditions. There are three conditional statements.

```
1.  IF condition THEN
    statement(s)
    END

2.  IFERR THEN
    statement(s)
    END

3.  IFLMD n THEN
    statement(s)
    END
```

If the condition specified is true, the statements between the THEN and END statements will be processed. If the condition is false, these statements will be ignored.

The conditions used in the first type of statement are ASCII character strings which are defined by the person writing the batch file and used as switches by the operator. For example, suppose a person is writing a batch file for running UPDAT and doing some file operations. If a part of the process requires the presence of an RX02 on the system, there would be a need for the operator using the batch file to be able to specify whether or not there was an RX02 present. The batch file writer would define a conditional section of the file as shown below.

```
IF RX02 THEN
statement(s)
END
```

The condition "RX02" is now used by the operator as a switch to the Chain command:

```
C FILE/RX02
```

The monitor stores the string of characters for comparison with the conditions in the batch file. There is only one pre-defined switch and the writer is free to create any other he or she desires. The pre-defined switch is "/QV' (quick ver'fy) which causes diagnostics to be run one pass only. Any number of switches may be used in a command.

The second type of conditional statement can be used with DRS-type diagnostics only. If a test error was detected by the last DRS-type diagnostic that was run in the batch file,the statements will be processed.

The third type of conditional uses the media-type byte in physical location 41. If the type code matches the one specified in the conditional ("n"), the statements will be processed.

8.2.2 GOTO

The GOTO statement is used to transfer control within a batch control file. When the monitor encounters a GOTO statement, it searches for the specified tag and resumes the batch process at the statement following the tag. A tag is an alphanumeric string terminated by a colon (":"). The tag may occur before or after the GOTO statement in the batch file. The following are examples of the GOTO statement:

```
      TAG1:
      R PROG1
      GOTO TAG1


      GOTO TAG2
           .
           .
           .
      TAG2:
      R PROG5
```

In the first example, the batch process will loop backwards until interrupted by the operator. In the second example, control will be transferred forward to TAG2. Any statements between the GOTO statement and the tag are ignored.

8.2.3 PRINT

The PRINT statement is used to force the typing of a line of text while typing is inhibited by the QUIET statement.

The format of the statement is:

K13

```
                PRINT text
```

The text on the same line as the PRINT will be typed.

8.2.4 QUIET

The QUIET statement is used to control typing of the batch file. The statement is used like a ""flip-flop". The first time the statement is encountered, all typing is suppressed (with the exception of error messages). The next time it is encountered, typing is reenabled. The third time it is encountered, typing is inhibited again and so on.

8.2.5 QUIT

The batch job is immediately stopped when a QUIT statement is encountered. The monitor returns to normal operator mode.

8.2.6 SMI/CMI

The SMI and CMI statements are used to enable and disable manual intervention modes in DRS-type diagnostics. Normally all testing that requires manual intervention by an operator are inhibited during batch control operations. These statements allow this to be over-ridden. Obviously caution is suggested when using this feature.

```
     SMI - set (allow) manual intervention
     CMI - clear (don't allow) manual intervention
```

CMI is the default state when a batch job is started.

8.2.7 WAIT

When a WAIT statement is encountered, the monitor stops processing the batch file and waits for the operator to type a CTRL-X (typed by depressing the CTRL and X keys together). This is typically used in conjunction with manual intervention feature as shown in the example below.

```
    PRINT THE NEXT DIAGNOSTIC REQUIRES THAT A
    PRINT SCRATCH MEDIA 1 BE MOUNTED IN THE RL01/02.
    PRINT TYPE ↑X WHEN READY
    WAIT
    R ZRLA??
```

8.3 Comments

Comments can be included in the batch file. Comments are typed as the file is processed unless QUIET mode is invoked. A comment is a string of text that starts with a semicolon.

```
    ;THE NEXT PROGRAM TESTS THE DZ11
    ;
    R ZDZB??       ;RUN THE DIAGNOSTIC
```

8.4 Batch Control of Diagnostics

For the purposes of batch control, there are two types of diagnostics: chainable non-DRS-type diagnostics and DRS-type diagnostics. The first type can be batched by a simple run command:

```
                  R DIAG[/n]
```

where n is an optional argument that specifies the number of passes that the diagnostic will run. The default is one pass.

DRS-type diagnostics require complete batch control. All commands normally entered by an operator must be in the batch file. For example:

```
      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]
```

This is just a short example. The concept to note is that the batch file is an INDIRECT COMMAND file for DRS. All commands that are required when running under operator control are necessary in the file. 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.

The user does not have to enter all commands via the batch file however. By using the SETUP utility (Chapter 6), all hardware and so..ware information could be supplied to the diagnostic prior to running the batch job. This is the recommended method for using DRS-type diagnostics in the batch control en ironment. If you preset all the parameters, the following commands are all that are necessary:

```
      R DIAG2
      START/PASS:n
      N
      N
      EXIT
```

where n is the number of passes to execute.

8.4.1 Diagnostic Abort

A DRS diagnostic in a chain file can be aborted by and ↑C or a ↑Z. If you abort with a ↑C, the chain file can test for the ↑C and alter the flow of control. For example:

```
         R DIAG
         N
         N
         EXIT
         IF  ↑C THEN
         GOTO L1
         END
```

If you abort with a ↑Z, the chain file cannot test for it and the flow of control cannot be changed.

8.4.2 Batch Control of Utilities

Most of the XXDP utilities can be used under batch control. The utilities thet are batch controlable are: UPDAT, SETUP, and PATCH. To run a utility under batch control, simply create a batch file that contains all of the commands that are normally entered by an operator. For example, to build an RX01 floppy diskette for XXDP using UPDAT under batch control:

```
         R UPDAT
         LOAD HMDX??.SYS
         SAVM DX0:
         CREATE DY0:
         FILE DY0:=DRSXM.SYS
         FILE DY0:=DIR.SYS
         FILE DY0:=DY.SYS
           EXIT
```

Note that the dialogue with UPDAT must end with an EXIT command in order to finish the batch job or to allow further batch functions.

APPENDIX A

GLOSSARY

This is a glossary of common terms used in connection with XXDP.

autodelete - a possible effect of the file transfer process whereby a file from the input medium replaces a file of the same name on the output medium. In UPDAT, only transfers initiated by a FILE command can result in autodeletion.

boot block - the first physical block on a medium (block zero). This block contains the XXDP secondary bootstrap for the device.

bootstrap - very simple code used to load and start more complex code from a medium 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, usually from a file, as opposed to executable code

console terminal - the video or hardcopy terminal attached to the system via the DL interface at bus address 177560.

device driver - that software which has the function of controlling the operation of a specific hardware component in a system. An RX02 driver, for example, is that software that accomplishes such tasks as selecting a physical block, reading a block of information, etc. 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 medium.

edit to modify text information in a file ed'tor - a utility program used to modify text files

hardware table - data structure where DRS stores the information regarding units being tested

load - the process whereby the contents of a file containing a program image are placed in memory.

med'um - phys'cal storage such as a disk or magtape. In this manual, the term "medium" is equivalent to "XXDP medium".

pass - a unit of diagnostic operation. A DRS-type diagnostic pass is defined to be 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 medium.

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").

pr'mary bootstrap - code, usually stored in a ROM, which loads the "boot block" (block 0) from a medium 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 medium. This code is loaded and started by the primary bootstrap and 'n turn loads an starts the XXDP monitor.

software table - data structure where DRS stores information regarding operational characteristics of a diagnostic

sw'tch - a modifier for a command

system med'um - the medium on the device from which the XXDP System was booted

text a collect'on of ASCII formatted data consisting of printing characters, tabs, carriage returns and form feeds.

virtual location - a relative mem reference. A program image that has been loaded int : program buffer by UPDAT uses virtual locations; that program location 0 is not physical memory location 0, it is the first physical memory location in the program buffer. The XXDP monitor does absolute loads and in this case program location 0 is not virtual, but 's actually memory location 0.

XXDP medium - physical storage, such as a disk pack, MAGtape, cassette, etc., that has been formatted for XXDP use.

APPENDIX B

COMMAND SUMMARY

This appendix summarizes the following:

- Monitor Commands
- DRS Commands, Switches, and Flags
- UPDAT Commands
- PATCH Commands
- XTECO Commands
- Batch Control Functions

B.1 Monitor Commands

B.1.1 SM Monitor Commands

C	run a batch job (chain)
D	list directory of load medium
E	enable alternate system device
F	set the terminal fill count
H	type help information
L	load a program
R	run a program
S	start a program
TEST	run batch file: SYSTEM.CCC

B.1.2 XM Monitor Commands

Boot	Boot a device
Chain	run a batch job (chain)
COpy	Copy a file or device
DAte	Set the date or report the date
DElete	Delete a file
Directory	list directory of load medium
Enable	enable alternative drive for system device
Help	type help information
Initial'ze	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 dev'ce or system parameter
Start	start a program
Type	Type a file

B.2 DRS Commands

ADD	activate a unit for testing
CONTINUE	continue diagnostic at test that was interrupted by a ↑C
DISPLAY	print a list of device information
DROP	deactivate a unit
EXIT	return to XXDP runtime monitor
FLAGS	print status of all flags
PRINT	print statistical information
PROCEED	continue from an error halt
RESTART	start diagnostic and do not initialize
START	start the diagnostic and initialize
ZFLAGS	reset all flags

B.3 DRS Command Switches

/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 the tests specified
/UNITS:unit-list	command will affect only specified units

B.4 DRS Flags

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
IDR	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)

B.5 UPDAT Commands

ASG	assign a logical name to a device
BOOT	bootstrap a device
CLR	clear UPD2 program buffer
COPY	copy entire medium
DEL	delete a file or files
DIR	give directory of specified medium
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 the runtime monitor
FILE	transfer a file or files
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 or files
PRINT	print a file on the line printer
READ	read a file to check validity
REN	rename a file
SAVM	save a monitor on a disk
SAVE	save a monitor on a tape
TYPE	type a file on the console terminal
XFR	set transfer address
ZERO	initialize a medium

B.6 PATCH Commands

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	Crreate patched file
SAVP	Save input table
TYPE	Print input table on terminal

B.7 XTECO Commands

B.7.1 Non-edit commands

```
EDIT - modify a file
EXIT - return to monitor
PRINT - print a file on the line printer
TEXT - create new text file
TECO - modify a file on disk
TYPE - type a file on the console terminal
```

B.7.2 Edit Commands

```
A  - append text to that currently in memory
C  - move the pointer character by character
D  - delete character(s)
EX - exit edit mode
I  - insert text
J  - move the pointer to the beginning of text
K  - delete line(s)
L  - move the pointer line by line
N  - search for specified string in remainder of text file
S  - search for specified string in text now in memory
T  - type text
ZJ - move the pointer to the end of text
```

B.8 Batch Control Functions

Monitor Commands	monitor commands: R, L, S, C, and E)
Utility Commands	UPDAT, SETUP, etc.
DRS Commands	all DRS commands and diagnostic dialogue
conditionals	sections of the batch file can be processed conditionally under operator control or runtime conditions (e.g. IF, THEN)
GOTO tag	begin processing at another section of the batch file designated by "tag"
QUIET	inhibit printing of batch file if printing or enable printing if printing was inhibited previously
PRINT	temporary override of QUIET
SMI/CMI	enable/disable manual intervention operations in specialized diagnostics
QUIT	terminate the batch operation
WAIT	stop batch operation until the operator types a Control X

APPENDIX C

DEVICES SUPPORTED

XXDP supports most mass storage devices. The following table lists all devices supported, the mnemonic used to specify the device and the name of driver the file.

Device	Mnemonic	Driver	DISTRIBUTION MEDIA
RP04/5/6	DB	DB	
TU58	DD	DD	TU58
RL01/2	DL	DL	RL01 / RL02
RK06/7	DM	DM	RK06 / RK07
RM02/3	DR	DR	
RX02	DY	DY	RX02
PRINTER	LP	LP	
TM02	MM	MM	800 BPI MT
TS04	MS	MS	1600 BPI MT
UDA50	DU	DU	
RD/RX	DU	DU	

All drivers assume that the CSR address for the device is the standard address as given in the Peripheral Handbook. If you have a system with a device at a non-standard address, you can modify location 12 in the driver using UPDAT.

XXDP device drivers are, by necessity, small and limited in function. They can detect and report three types of errors: read, write and hard. These errors are reported and control is returned to the utility being used. The utility then takes any further action required. Since the functionality of the drivers is limited, the user is required to run diagnostics on the device in question if an error persists.

APPENDIX D

COMPONENT NAMES

All the component file names begin with the letter H. The second letter in the name determines the component type, as follows:

M --	monitors
U --	utility programs
D --	device drivers
S --	runtime services
Q --	manuals

The third and fourth letters indicate the component name and the last two letters are the version numbers.

The utility programs are distributed under their common names, like UPDAT and XTECO, for user convenience and have .BIN (or .BIC) extensions.

Description	File Name	Release Name
MONITOR files		
XXDP V2 RESIDENT MONITOR	XXDPSM.SYS	HMSM??.SYS
XXDP V2 EXTENDED MONITOR	XXDPXM.SYS	HMXM??.SYS
XXDP V2 DATE UTILITY	DATE.SYS	HUDA??.SYS
XXDP V2 DIRECTORY UTILITY	DIR.SYS	HUDI??.SYS
XXDP V2 HELP FILE	HELP.TXT	HQLP??.TXT
DRIVER files		
RP04,5,6 DRIVER & BOOT	DB.SYS	HDDB??.SYS
TU58 DRIVER & BOOT	DD.SYS	HDDD??.SYS
RL01/2 DRIVER & BOOT	DL.SYS	HDDL??.SYS
RK06/7 DRIVER & BOOT	DM.SYS	HDDM??.SYS
RM02/3 DRIVER & BOOT	DR.SYS	HDDR??.SYS
MSCP DRIVER & BOOT	DU.SYS	HDDU??.SYS
RX02 DRIVER & BOOT	DY.SYS	HDDY??.SYS
PRINTER DRIVER	LP.SYS	HDLP??.SYS
TM02 DRIVER & BOOT	MM.SYS	HDMM??.SYS
TS04/TS11 DRIVER & BOOT	MS.SYS	HDMS??.SYS
UTILITY files		
XXDP V2 DECX11 CONFIGURATOR AND LINKER	DXCL.BIC	HUXC??.BIC
XXDP V2 UPDATE UTILITY	UPDAT.BIC	HUP2??.BIC
XXDP V2 PATCH UTILITY	PATCH.BIC	HUPA??.BIC
XXDP V2 SETUP UTILITY	SETUP.BIC	HUSU??.BIC
XXDP V2 XTECO UTILITY	XTECO.BIC	HUTECO.BIC
UFD files		
XXDP V2 UFD MENU UTILITY	MENU?0.BIC	HUMEB0.SYS
DRS files		
XXDP DIAG SUPR SML	DRSSM.SYS	HSAA??.SYS
XXDP DIAG SUPR EXT	DRSXM.SYS	HSAX??.SYS
Documents		
XXDP FILE STRUCT DOC	N/A	CHQFSB0.DOC
XXDP V2 FILE STRUCT DOC	N/A	CHQ???0.DOC
XXDP DRIVER PROGR GUIDE	N/A	CHQ????.DOC

APPENDIX E

BUILDING XXDP

E.1 Monitor and Required Files

The minimum files that must be put on a bootable XXDP medium are the monitor for that medium, the device driver for that medium, the DRS (file name: DRSS.SYS or DRSX.SYS) and the directory utility (file name: DIR.SYS). The monitor file (file name:) must be loaded by UPDAT and then saved on the medium using either the CREATE command. These commands are described in Chapter 4.

The remaining files may be put onto the medium using any UPDAT file transfer commands.

Examples (RX02 and TS04):

```
.R UPDAT

CREATE DY0:

FILE DY0:=DY.SYS

FILE DY0:=DIR.SYS

FILE DY0:=DRSXM.SYS

EXIT

.R UPDAT
```

```
CREATE MS0:

PIP MS0:=MS.SYS

PIP MS0:=DIR.SYS

PIP MS0:=DRSXM.SYS

EXIT
```

The process described above places the MINIMUM XXDP system on a medium. You can add as many other system components (drivers and utilities) as you wish. Don't forget to modify location 370 in the XXDPXM monitor if you are using a system that is on 50Hz power. (Location 370 must contain a zero value for 60Hz and a non-zero value for 50Hz.)

E.2 Update Kits

The file XXBLD.CCC is a batch control file that updates/builds XXDP media automatically. To start the file, use the chain command. The file accepts switches that specify the media type to build and the mode in which to build. All supported XXDP media may be built. The media being built are always assummed to be mounted in drive (unit) 0 of the device and that the drive is ready and write-enabled.

The format of the command line for starting the build process is:

```
        C XXBLD/device[/mode]
```

device - the mnemonic for the device to be built/updated. Supported devices and their mnemonics are listed in appendix C. If no device is specified, a short help message will be printed.

mode - manner in which to build/update. Available modes are:

DRIVER	a bootable medium with all XXDP drivers
MONITOR	a bootable medium with all XXDP monitors
UTILITY	a bootable medium with all XXDP utilities
SYSTEM	a combination of the above three modes

If no mode is specified, a bootable medium is built. A bootable medium consists of a bootable monitor image, the runtime services, the directory utility, the driver for the medium and UPDAT.

Except in the case of sequential devices (e.g.; magtape), the medium is not changed except for the replacement/addition of the new XXDP components specified by the mode switch. You may want to back up files that are critical however. Sequential media are destroyed. A warning message is produced and you are given the opportunity to abort the process.

To obtain help while running the update batch job, use the following command:

```
        C XXBLD/HELP
```

APPENDIX F

USER TIPS

F.1 DRS Table Building

To save time and energy, prebuild the hardware and software tables in a diagnostic using the SETUP utility. Customize the files for a specific system on the XXDP medium for that system or customize files for several systems on medium shared between systems. Remember, you can always change the tables on the fly by using the START command or permanently change the files by using SETUP.

Another way to make XXDP work for you is to use the batch control functions described in Chapter 8 of this manual. 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, or systems, you are working with. Then write a batch control file that implements the various modes based on conditionals. This allows you to enter one command to XXDP and then let the system do the rest.

A simple example of this type of batch control file is follows. For the purposes of this example, we will use a fictional diagnostic called 'DIAG1". The normal operator dialogue with this diagnostic (with hardware tables already built) is:

```
        .R DIAG1
        DR>STA
        CHANGE HW (L) ?  N
        CHANGE SW (L) ?  Y
        TEST ALL SECTORS (L) ?
```

The batch file that has been created for this diagnostic is called 'DISK.CCC' and is listed below.

```
IF QV THEN
R DIAG1
STA/PAS:1
N
Y
N
END
IF REPAIR THEN
R DIAG1
STA/FLA:LOE
N
Y
Y
N
END
```

This file defines two test modes: quick verify and repair. Note how the batch file manipulates the software questions and also avoids answering the hardware questions since the tables were already created. The user invokes either of the two modes with one of the following command

```
                C DISK/QV
```

or,

```
                C DISK/REPAIR
```

APPENDIX G

ERROR MESSAGES

The error messages that could occur during an operation are listed alphabetically below. The lower case letters "device error" refer to a series of specific errors that can be detected by the device drivers. The possible errors are listed after the error messages.

```
? ADDRESS NOT FOUND              (PATCH)
       The specified address does not exist as an entry in
       the input table.

? BAD ADDR                       (Monitor)
       An invalid address was specified with a run or start
       command.

? CHECKSUM ERROR                 (Monitor, UPDAT, PATCH)
       Each block in a binary file has a checksum stored in
       it.  If the checksum calculated while reading the
       block does not match the checksum in the block, this
       error is printed.  Try the operation again, but the
       file was probably corrupted.

? COMMAND NEEDS ARGUMENT         (PATCH)
       The command typed by the operator requires an
       argument, but none was given.

? DELETE OLD FILE                (PATCH)
       The specified output filename already exists.

? DEVICE BOOT BLOCK NOT INITIALIZED     (Monitor)
       An attempt to boot a device that does not contain an
       initialized boot block will be reported as shown and
       the device will not be booted.

? device error                   (Monitor, UPDAT)
```

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. This program appends any additional information as shown in the device error messages. The errors that can be reported by drivers are: READ ERROR and WRITE ERROR.

? device error DIRECTORY (PATCH)
The specified device error occured 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 non-transfer operation).

? device error ON INPUT DEVICE (Monitor, UPDAT,PATCH)
The specified 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 (Monitor, UPDAT)
The specified 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 (Monitor, UPDAT,PATCH)
The specified 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 OUTPUT DEVICE DIRECTORY (Monitor, UPDAT)
The specified 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 (Monitor,UPDAT,PATCH)
The specified error occurred while the driver for the specified device was being loaoed into memory.

? device error WHILE READING filename
(Monitor,UPDAT,PATCH) The specified error occurred while the specified file was being read.

? DEVICE FULL (Monitor, UPDAT)
The capacity of the output device has been exceeded. For disk devices (random access), there are not enough physical blocks remaining to store the file. Any blocks allocated during the attempt to write the file are deallocated. For tape devices (sequential access), 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 medium.

? DIRECTORY FULL (Monitor, UPDAT)
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 medium.

? END-OF-MEDIUM (PATCH)
While reading a file, the end of the file was encountered before it was expected.

ERR HLT (DRS)
A halt-on-error has occurred. DRS is now back at command mode. Halt-on-error only occurs if operator has specified this mode with a flag to DRS.

? FILE ALREADY EXISTS (Monitor,PATCH)
The name of the file specified for output matches that of a file that already exists on the output medium. Delete the old file or use a different name.

? FILE NOT FOUND (PATCH)
The specified input filename does not exist.

? FILE TOO BIG (PATCH)

ILL INTER nnn PC nnnnnn Ps nnnnnn (DRS)
An unexpected interrupt occurred through vector nnn. The Program Counter and Processor Status Word at the time of the interrupt are given.

? INPUT TABLE EMPTY (PATCH)
The specif'ed command cannot be executed because there are no entr'es in the input table.

```
? INPUT TABLE FULL                              (PATCH)
        The input table is full and cannot accept any more
        entries.

  INSUFF MEM                                    (DRS)
        There is not enough memory space to store table
        information for the number of units that the user
        wants to specify.

  INVAL SWTCH FOR CMND                          (DRS)
        The user specified a non-existent or non-applicable
        switch in the previous command.  Refer to the table
        of switches Chapter 3.

  INVAL UNIT                                    (DRS)
        The user specified a unit that does not exist.

? INVALID ADDRESS                               (PATCH)
        An address given in the last command was not legal
        (possibly an odd number).  Check the command and
        re-enter properly.

? INVALID ADDRESS                               (UPDAT)
        There are three operations that can cause this
        error.  One, when using the MOD command to modify a
        virtual location, the address of the location given
        was odd or not within the upper and lower core
        limits.  Two, the address given in a LOCORE command
        was higher than the current high core limit.  And,
        three, the address given in a HICORE command was
        lower than the current low core limit.

? INVALID COMMAND                 (Monitor,UPDAT,PATCH)
        You have entered a command that is not recognizable.
        Check the command (especially spelling) and re-enter
        properly.

? INVALID DEVICE                  (Monitor, UPDAT,PATCH)
        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 non-tape
        device.  For COPY, the specified devices are not
        identical types.

? INVALID FILENAME                (Monitor, UPDAT, PATCH)
        You specified a filename in the previous command
        that was not in the proper format.
```

```
? INVALID MODULE NAME                             (PATCH)
      A DEC/X11 module name was incorrectly specified.

? INVALID NUMBER         (Monitor, UPDAT, PATCH)
      A number specified in the last command was not
      entered properly.  Possible problems are:  not a
      number (e.g., 12e4) or not proper radix (e.g., 1292
      is not octal).

? INVALID SWITCH                  (Monitor, UPDAT)
      The last command was entered with a switch that is
      not recognized by UPDAT.  Re-enter the command
      properly.

? LOGICAL DEVICE NOT ASSIGNED           (Monitor, UPDAT)
      An attempt was made to use a logical unit number
      without first assigning it to a device.  (See ASG
      command.)

  LOOKUP ERROR filnam                            (DRS)
      This error message actually comes from the XXDP
      monitor.  If the file name is DRSXM.SYS or
      DRSSM.SYS, then the diagnostic being run requires
      the DRS, but the DRS file is not on the system
      medium.  Any other file name indicates that the
      diagnostic attempted to open a file that does not
      exist on the system medium.

  LOOP CHNG                                      (DRS)
      The range of the loop changed while looping on error
      was in progress.

? MEMORY ERR AT LOCATION:  xxxxxxxx              (Monitor)
      A nonexistent memory or a memory parity error
      occurred at the location specified.  If the type of
      error is detected that will also be reported.

      Example:
       ? MEMORY PARITY ERR AT LOCATION: xxxxxxxx

? MODULE NAMES NOT ALLOWED WITHOUT MAP FILE      (PATCH)
      The operator attempted to specify a module name in
      the MOD command without first loading the proper MAP
      file.

? MODULE NAME NOT FOUND                 (PATCH)
      The specified module name does not exist within the
      DEC/X11 runtime exerciser.
```

```
? MUST BE EVEN                                   (PATCH)
      The operator attempted to specify an odd number as
      an address.

? MUST BE OCTAL                           (PATCH)
      The operator attempted to type a non-octal number.

? NEED NUMBER                                    (PATCH)
      The operator omitted a numeric value from a command
      that expected one.

? NO DEVICE DEFAULTS                             (PATCH)
      Default device names are not allowed.

  NO UNIT                                        (DRS)
      There are no active units.  Either no units have
      been specified or all units have been dropped.

? NOT ENOUGH ROOM TO LOAD DRIVER                 (PATCH)
      The driver for the specified device will not fit
      into memory.

? NOT FOUND                               (UPDAT, PATCH)
      The file specified for input in the last command was
      not found on the device specified.

? NOT FOUND:  XX.SYS                      (UPDAT)
      The driver for device "xx" was not found on the
      system medium.  This message is printed by the
      monitor driver which is used to load device drivers
      for UPDAT.  Transfer the required driver file to the
      system medium.

? NOT FOUND:  filnam                             (Monitor)
      The specified file was not found by the monitor.
      The monitor can only read files that are on the
      system medium unless a specific device is specified.

? NOT FOUND:  xx.SYS                             (Monitor)
      The driver for device "xx" was not found on the
      system medium.  Transfer the required drive file
      (XX.SYS) to the system medium.

  NOT HALTED                                     (DRS)
      The user attempted to enter a PROCEED command when
      the DRS had not executed a "halt-on-error" sequence.

? NUMBER TOO BIG                                 (PATCH)
      The value typed was too large for its intended
      purpose.

?  OPTION MODULE NAME NOT FOUND                   (PATCH)
      The specified option module does not exist.

?  OVERFLOW                               (Monitor, UPDAT)
      An attempt was made to load too large a program into
      the program buffer.

   PASS ABORTED FOR THIS UNIT                      (DRS)
      Testing was prematurely ended for the current unit
      being tested.  There is usually an error message
      from the diagnostic given prior to this message.
      Refer to the specific diagnostic documentation for
      the reason that the unit may have been aborted.

?  READ ERROR                                  (Monitor)
      A device error occured while reading.  Retry the
      operation.

?  SPECIFY DEVICE                  (Monitor, UPDAT, PATCH)
      The last command specified does not allow the use of
      default device specifications.  Re-enter the command
      with the device(s) explicitly specified.

?  SYNTAX ERROR                           (Monitor, UPDAT,
      PATCH)
      The last command was entered improperly.  Re-enter
      the command properly.

   TRAP ERR AT:  nnnnnn                            (DRS)
      An unrecognized TRAP instruction was executed.  The
      TRAP instruction is used to communicate between the
      diagnostic and DRS.  This error should never occur
      in field operation.  Please report the problem if it
      does.

   TST   TOO BIG                                   (DRS)
      The user specified a test number that is larger than
      the number of tests in the diagnostic program being
      run.

?  UNEXPECTED END-OF-FILE         (Monitor, UPDAT,PATCH)
      The logical end of a file was encountered before it
      was expected.  The file in question is corrupt.

?  UNEXPECTED INTERRUPT FROM:  xxxxxxxx        (Monitor)
      An unexpected interrupt occurred from the indicated
```

location.

? WRITE ERR (Monitor)
A device error occurred while writing. Retry the operation. If the error as reported from the driver contains additional information this will also be reported by appending this information to the end of the above report. Some errors that be detected and easily corrected by the user would be the case where the device is write locked or does not have a write ring in stalled.

Example:
```
 ? WRITE ERR - DK0: WRITE LOCKED
```

?WRONG MAP FILE FOR MONITOR TYPE (PATCH)
The MAP file in memory does not have the specified monitor type.

INDEX

[END OF MANUAL]