

HEWLETT-PACKARD



MS-DOS User's Reference

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Vectra is a U.S. registered trademark of Hewlett-Packard Company.

MS-DOS is a U.S. registered trademark of Microsoft, Incorporated.

**Personal Office Computer
Division
974 E. Arques Avenue
P.O. Box 486
Sunnyvale, CA 94086, U.S.A.**

To order this manual, specify Part Number 45951-90003

Printing History

First Edition—September, 1985

Printed in Singapore

Vectra MS-DOS User's Reference Manual



**Manual Part No.
45951-90003**

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

FCC Statement

Federal Communications Commission Radio Frequency Interference Statement

Warning: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

More About Radio and Television Interference

Because the HP Vectra PC generates and uses radio frequency energy, it may cause interference with radio and television reception in a residential installation.

Hewlett-Packard's system certification tests were conducted with HP-supported peripheral devices and HP shielded cables, such as those you receive with your system. HP Vectra meets the requirements for a Class B computing device in accordance with the specifications of Part 15, Subpart J, of FCC rules. These rules are designed to provide reasonable protection against interference with radio and television reception in a residential installation.

Hewlett-Packard provides instructions for using this computer in manuals covering setup, connection of peripheral devices, operation, service, and technical reference.

Installing and using the computer in strict accordance with Hewlett-Packard's instructions will minimize the chances that the HP Vectra will cause radio or television interference. However, Hewlett-Packard does not guarantee that the computer will not interfere with radio and television reception.

If you think your computer is causing interference, turn it off to see if the radio or TV reception improves. If the reception:

- Does not improve, your computer is not causing the problem.
- Does improve, your computer is causing the problem.

To correct interference, take one or more of the following steps:

- Relocate the radio or TV antenna
- Move the computer away from the radio or television.
- Plug the computer into a different electrical outlet, so that the computer and the radio or television are on separate electrical circuits.
- Make sure that all of your peripheral devices are certified Class B by the FCC.
- Make sure you use only shielded cables to connect peripheral devices to your computer.
- Consult your computer dealer, Hewlett-Packard, or an experienced radio/television technician for other suggestions.
- Order the FCC booklet called *How to Identify and Resolve Radio-TV Interference Problems* from the U.S. Government Printing Office, Washington, D.C. 20402. The stock number of this booklet is 004-000-00345-4.

Warning:
Electrical Safety

For the user's safety, the power cords supplied with this product have grounded plugs. The power cords should be used with properly grounded (3-hole) wall outlets to avoid electrical shock. (You can also use multiple-outlet strips that have their own circuit breakers.)

Table of Contents

Chapter 1:

Introduction

What is MS-DOS?	1-1
HP Vectra Documentation	1-2
What is in This Manual	1-3

Chapter 2:

Understanding MS-DOS Concepts

Understanding Files	2-1
What Is A File?	2-1
Data And Program Files	2-2
How To Name Your Files	2-3
Invalid And Reserved File Names	2-4
MS-DOS Reserved File Names	2-8
Wildcard Characters	2-10
File Name Summary	2-12
File Attributes	2-13
Root Directories, Subdirectories, and File Allocation Tables	2-14
Directories	2-14
Root Directory	2-15
Subdirectories	2-15
File Allocation Table (FAT)	2-16
Active Drive and Current Directory	2-17
Active Drive	2-17
Current Directory	2-18
MS-DOS Paths	2-19
Files and Paths	2-20

Chapter 3:**MS-DOS Command Basics**

What is an MS-DOS Command?	3-1
Internal Commands	3-1
External Commands	3-1
The MS-DOS Command Line	3-2
The MS-DOS System Prompt	3-2
Constructing A Command Line	3-3
Executing A Command Line	3-5
MS-DOS Commands	3-5
File-Oriented Commands	3-6
Disc-Oriented Commands	3-8
Peripheral Commands	3-10
System Commands	3-11

Chapter 4:**Batch Processing**

What is Batch Processing	4-1
Creating a Batch Command Line	4-2
Executing a Batch Command File	4-3
Chaining Batch Command Files	4-4
AUTOEXEC Batch File	4-5
Batch Command Files With Replaceable Paramaters ..	4-6
Using %0 - %9	4-6
Using the SET Command	4-8
Batch Commands	4-9
ECHO	4-10
FOR	4-12
GOTO	4-14
IF	4-16
PAUSE	4-18
REM	4-20
SHIFT	4-21

Chapter 5:**Using the MS-DOS Keyboard**

Editing a Command Line	5-1
Printing the Screen.....	5-4
Other Useful Functions	5-5
Unused Keys	5-7

Chapter 6:**Hard Disc Preparation**

Hard Disc Basics	6-1
Partitioning	6-2
FDISK	6-3
Creating An MS-DOS Partition	6-4
Changing The Active Partition	6-7
Deleting An MS-DOS Partition	6-8
Displaying Partition Information	6-9
Using FDISK With Drive D:.....	6-9
Formatting a Partition and Transferring MS-DOS and PAM	6-10

Chapter 7:**System Configuration**

Overview	7-1
Configuration Commands	7-2
BREAK	7-3
BUFFERS	7-4
COUNTRY	7-6
DEVICE	7-8
FCBS	7-9
FILES	7-11
LASTDRIVE	7-13
SHELL	7-14
Device Drivers	7-15
DEVICE Command	7-15
ANSI.SYS	7-17
VDISK.SYS	7-17

Chapter 8:**Standard Input and Output**

Standard Input and Output Devices	8-1
Redirecting Standard Input and Output	8-2
Piping Standard Input and Output	8-5
Filters	8-6
Tips on Using Redirection, Piping, and Filters	8-7

Chapter 9:**DEBUG**

Introduction	9-1
DEBUG Commands	9-1
DEBUG Command Parameters	9-4
Memory Addressing on the HP Vectra	9-7
How to Start DEBUG	9-10
DEBUG Command Descriptions	9-12
The Assemble Command	9-13
The Compare Command	9-16
The Dump Command	9-17
The Enter Command	9-19
The Fill Command	9-21
The Go Command	9-22
The Hex Command	9-25
The Input Command	9-26
The Load Command	9-27
The Move Command	9-29
The Name Command	9-30
The Output Command	9-33
The Procedure Command	9-34
The Quit Command	9-35
The Register Command	9-36
The Search Command	9-39
The Trace Command	9-40
The Unassemble Command	9-42
The Write Command	9-44
Debug Error Messages	9-46

Chapter 10:**MS-LINK**

What MS-LINK is and What it Does	10-1
Definition of Terms	10-4
How MS-LINK Combines and Arranges Segments ...	10-7
Files that MS-LINK Uses	10-12
Input Files	10-13
Output Files	10-14
Virtual Memory File (VM.TMP File)	10-14
Running MS-LINK	10-16
Working With MS-LINK	10-16
Using the Text-Prompt Method	10-17
Using the Command-Line Method	10-17
Using the Response-File Method	10-20
Creating a Response File	10-21
Command Prompts	10-23
Object Modules [.OBJ]:	10-24
Run File [First-Object-filename.EXE]:	10-24
List File [NUL.MAP]:	10-24
Libraries [.LIB]:	10-26
Command Characters	10-27
The Plus-Sign	10-28
The Semicolon	10-29
[CTRL]-C	10-29
Optional Switches	10-30
The /DSALLOCATE Switch	10-31
The /HIGH Switch	10-32
The /LINENUMBERS Switch	10-32
The /MAP Switch	10-32
The /PAUSE Switch	10-34
The /STACK: <number> Switch	10-34
Error Messages	10-36
MS-LINK Examples	10-42

Chapter 11:**EDLIN**

Introduction to EDLIN	11-1
How To Start EDLIN	11-2
Information Common To All EDLIN Commands ...	11-4
EDLIN Syntax	11-7
EDLIN Command Options	11-8
Summary	11-9
EDLIN Commands	11-9
The Append Command	11-11
The Copy Command	11-13
The Delete Command	11-16
The Edit Command	11-19
The End Command	11-22
The Insert Command	11-23
The List Command	11-26
The Move Command	11-29
The Page Command	11-31
The Quit Command	11-32
The Replace Command	11-33
The Search Command	11-37
The Transfer Command	11-41
The Write Command	11-42
EDLIN Error Messages	11-43

Chapter 12:**MS-DOS Command Descriptions**

Introduction	12-1
Command Descriptions	12-1
Syntax Notation	12-2
ASSIGN	12-6
ATTRIB	12-8
BACKUP	12-10
BREAK	12-13
CHDIR	12-15
CHKDSK	12-17
CLS	12-21
COMMAND	12-22
COMP	12-24
COPY	12-27
CTTY	12-32
DATE	12-34
DEL	12-36
DIR	12-38
DISKCOMP	12-42
DISKCOPY	12-45
ERASE	12-47
EXE2BIN	12-49
EXIT	12-51
FC	12-52
FIND	12-59
FORMAT	12-61
GRAFTABL	12-64
GRAPHICS	12-65
JOIN	12-67
KEYBUS	12-71
LABEL	12-73
MKDIR	12-75



MODE	12-77
Video Display (Options 1 and 2)	12-77
Serial Communication (Option 3)	12-79
Parallel Printer Port (Options 4 and 5)	12-80
MORE	12-83
PATH	12-84
PRINT	12-86
PROMPT	12-90
RECOVER	12-93
RENAME	12-95
RESTORE	12-96
RMDIR	12-98
SET	12-99
SHARE	12-101
SORT	12-103
SUBST	12-105
SYS	12-108
TIME	12-110
TREE	12-112
TYPE	12-114
VER	12-116
VERIFY	12-117
VOL	12-119

Appendix A:**MS-DOS Message Directory**

Disc and Device Errors	A-1
MS-DOS Command Messages	A-3

Appendix B:**Extended Screen and Keyboard Control**

Control Sequence Syntax	B-2
Control Sequences	B-3
Cursor Position	B-4
Cursor Up	B-5
Cursor Down	B-6
Cursor Forward	B-7
Cursor Backward	B-8
Horizontal and Vertical Position	B-9
Cursor Position Report	B-10
Device Status Report	B-11
Save Cursor Position	B-12
Restore Cursor Position	B-13
Erase in Display	B-14
Erase in Line	B-15
Set Graphics Rendition (SGR)	B-16
Set Mode	B-18
Reset Mode	B-20
Keyboard Key Reassignment	B-21

Appendix C:**Non-US Keyboard Layouts**

France/French	C-2
Germany/German	C-3
Italy/Italian	C-4
Spain/Spanish	C-5
U.K./English	C-6

Appendix D:

Disc and Disc Drive Compatibility

Appendix E:

MS-DOS Syntax Summary

Syntax Notation	E-1
Vectra MS-DOS 3.1 Commands	E-2

List of Figures

Figure No.	Figure Title	Page No.
2-1	MS-DOS File Names	2-3
2-2	MS-DOS Directory Structure	2-16
3-1	MS-DOS Command Line	3-3
6-1	Hard Disc Drives	6-2
8-1	Temporary Files Used During Piping	8-5
10-1	MS-LINK in Relation to Other System Elements	10-3
10-2	Segments and Groups in Memory	10-6

List of Tables

Table No.	Table Title	Page No.
2-1	Valid Characters For Filenames and Extensions	2-5
2-2	Invalid Characters for Filenames and Extensions	2-6
2-3	MS-DOS Predefined Extensions	2-7
2-4	MS-DOS Reserved Device Names	2-8
9-1	DEBUG Command Summary	9-2
10-1	How MS-LINK Handles Segments in Private, Public, and Common Types of Combinations	10-8
10-2	The Types of Files Used by MS-LINK	10-12
10-3	Default Extensions for Object and Library File names.	10-14
10-4	Default Extensions for Output File names	10-14
10-5	Summary and Syntax for the Three Ways to Run MS-LINK	10-17
10-6	Default Filename Extensions for Responses to Prompts	10-19
10-7	The MS-LINK Command Prompts and User Responses	10-24
10-8	The MS-LINK Optional Switch Functions	10-30

Introduction

What is MS-DOS?

HP Vector Documentation

What is in This Manual

1

Introduction

What is MS-DOS?

MS-DOS stands for "MicroSoft Disc Operating System", and it is the operating system available with the HP Vectra Personal Computer.

MS-DOS is the most widely used operating system for 16-bit personal computers. Literally thousands of applications are designed to run in the MS-DOS environment, enabling you to perform virtually any task with your HP Vectra system.

MS-DOS performs two vital functions for you:

- **Application Program Support.** MS-DOS is used by applications for basic data file operations and character input and output.
- **Disc, File, and System Management.** MS-DOS has a full range of commands and utilities to perform the disc and file management operations for your Vectra system and peripherals.

HP Vectra Documentation

This manual is part of a four-book documentation set. The set includes:

- *Setting Up Vectra*. This book explains how to install accessory cards and internal disc drives in your Vectra, and how to connect the monitor and keyboard.
- *Connecting Peripherals to Vectra*. This book explains how to connect printers, plotters, and other peripherals to your Vectra.
- *Using Vectra, DOS Version*. This book explains how to start Vectra and how to use its disc drives and input devices: the Keyboard, HP Mouse and HP Touch. It describes in detail how to use PAM — Vectra's user interface program, and also introduces novice users to some commonly-used MS-DOS commands.

The binder for this book contains the two discs that come with the system: the Vectra DOS 3.1 Disc, and the Vectra Supplemental Disc.

- *Vectra MS-DOS User's Reference*. This is the book you are reading. It contains a complete description of MS-DOS capabilities, commands, and utilities. MS-DOS concepts such as file and directory structure and batch command processing are explained. More details about the contents of this manual are provided later in this Introduction.

Three additional manuals discuss advanced software and hardware topics. Together, they comprise the Vectra programmer's tools:

- *Vectra MS-DOS Programmer's Reference*. This manual contains a description of the MS-DOS program interface, system interrupts, file structure, and programming procedures.

- *Vectra MS-DOS Macro Assembler*. This manual describes the MS-DOS 3.1 Macro Assembler, the Linker, and the DEBUG Utility.
- *Vectra Technical Reference Manual*. The Technical Reference Manual is a two-volume set:
 - Volume I: Hardware*, contains a complete description of the Vectra's hardware capabilities
 - Volume II: ROM BIOS*, provides a detailed description of the system's ROM BIOS structure and hardware interface.

What is in This Manual

This manual is directed to programmers and users who wish to use MS-DOS for system and application management. It is designed to be a reference tool, and assumes some familiarity with computer systems and terminology.

This User's Reference is divided into three sections, each of which addresses a different set of information.

SECTION 1. *Introducing MS-DOS Commands and Features*, describes basic MS-DOS concepts.

- Chapter 1. *Introduction*

- Describes all the manuals available for the Vectra System, and explains in detail what is in this manual.

- Chapter 2. *Understanding MS-DOS Concepts*

- Describes the MS-DOS concepts of files and directories.

■ Chapter 3. *MS-DOS Command Basics*

Covers creating an MS-DOS command line, and provides information you must know to use MS-DOS commands.

■ Chapter 4. *Batch Processing*

Complete documentation on MS-DOS Batch Command Processing. Topics include creating a batch command file, the AUTOEXEC file, and detailed descriptions of the Batch File commands.

■ Chapter 5. *Using the MS-DOS Keyboard*

Description of keyboard functionality in MS-DOS.

SECTION 2. *Advanced MS-DOS Features*, contains information about hard disc preparation, system configuration, Standard Input and Output, and the MS-DOS utilities DEBUG, LINK and EDLIN.

■ Chapter 6. *Hard Disc Preparation*

Contains detailed information on preparing the hard disc, including partitioning for multiple operating systems.

■ Chapter 7. *System Configuration*

The MS-DOS Configuration commands are described. You will also learn how to install device drivers to add system components.

■ Chapter 8. *Standard Input and Output*

Descriptions of the MS-DOS capabilities to redirect and pipe input and output, and the three MS-DOS filters, FIND, SORT, and MORE.

■ Chapter 9. *DEBUG*

Complete documentation on *DEBUG*, the program debugging tool provided by MS-DOS.

■ Chapter 10. *LINK*

Documents the MS-DOS object module linker, used primarily by programmers to link program modules into one program.

■ Chapter 11. *EDLIN*

This final chapter of Section 2 documents *EDLIN*, the MS-DOS line editor.

SECTION 3. *MS-DOS Command Descriptions*, the final section, contains the alphabetical reference guide to MS-DOS commands.

■ Chapter 12. *MS-DOS Command Descriptions*

The MS-DOS commands are listed in alphabetical order. Each command description includes the complete syntax, typical uses and examples, and complete notes.

APPENDICES. Supplementary information, including MS-DOS System and Error Messages, the ANSI device driver, and disc compatibility.

INDEX. An alphabetical listing of subjects with page number references.

2

Understanding MS-DOS Concepts

Understanding Files	2-1
What Is A File?	2-1
Data And Program Files	2-2
How To Name Your Files	2-3
Invalid And Reserved File Names	2-4
MS-DOS Reserved File Names	2-6
Wildcard Characters	2-10
File Name Summary	2-13
File Attributes	2-13
Root Directories, Subdirectories, and	
File Allocation Tables	2-14
Directories	2-14
Root Directory	2-15
Subdirectories	2-15
File Allocation Table (FAT)	2-16
Active Drive and Current Directory	2-17
Active Drive	2-17
Current Directory	2-18
MS-DOS Paths	2-18
Files and Paths	2-19

2

Understanding MS-DOS Concepts

In this chapter we will discuss basic MS-DOS concepts. This material, as well as that contained in the chapter titled *MS-DOS Command Basics*, provides the foundation for our later discussions of MS-DOS. This chapter contains information on MS-DOS files and file names, directories, and subdirectories. Several MS-DOS commands directly related to these topics are discussed.

Understanding Files

A cursory examination of the MS-DOS commands will reveal a decided bias towards files and discs. File and disc management is at the heart of MS-DOS. We will therefore begin this introduction to MS-DOS and its capabilities by exploring files.

What Is A File?

Information contained on your system's discs is stored in files. A disc is like a large filing cabinet. It has the capability of storing large amounts of information. However, that information cannot be simply "dumped in the drawers". It must be stored in a manner that is easy to retrieve.

The MS-DOS file structure provides this organization, while at the same time meeting the sometimes diverse requirements of different types of data. Files can be as small as one character or as large as 32 million. In addition, diverse types of data may be stored using the same file system.

Data And Program Files

Most application programs use files to store information. Word processors store text in files, spreadsheets store numbers and equations, accounting programs store financial data, etc. In short, data files are an integral part of almost any application program.

The actual format of the data within the file varies from application to application. MS-DOS does not concern itself with the contents of the files. Its commands are designed to operate on the files themselves. Thus, a file containing spreadsheet data and one containing word processing text are indistinguishable to MS-DOS.

A program is a set of instructions which tells the computer how to perform a certain task. Programs are also stored on discs in files.

Therefore, during the remainder of the Manual, when a file is referred to, it can be either a data or program file, unless stated otherwise. The design of the MS-DOS file structure allows both types of files to be treated almost identically.

How To Name Your Files

Whether a file contains data or a program, it must have a name. A name consists of two parts; a filename and an optional extension. The filename can be up to 8 characters in length, the extension up to 3 characters. The filename and extension must be separated by a period (.). A file name can't consist of an extension only, such as .BAS; if an extension is used, it must be preceded by a filename. Figure 2-1 outlines the requirements of a file name.

Note



Throughout the remainder of the manual, frequent references will be made to files. The term File Name (two words) will be used to describe a Filename (one word) and Extension. There is a difference between the two, and you should make careful note of whether a File Name or Filename is being referred to.

FILENAME . EXT

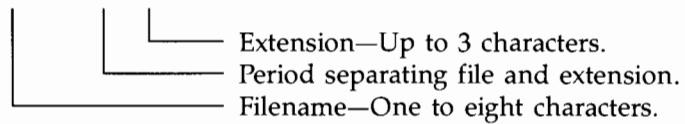


Figure 2-1. MS-DOS File Names

Proper file naming is the heart of organizing data on your disc. Many applications will name their data files without consulting you. For example, most accounting programs name their data files internally, rather than asking the user for a name. Some programs ask the user to provide a filename, then append a specific extension automatically. Other applications, such as word processing, place the entire responsibility of selecting a file name with the user.

Invalid And Reserved File Names

Although the MS-DOS file naming rules allow a virtually unlimited range of filenames and extensions, there are certain characters, filenames and extensions which can't be used. These are used elsewhere by MS-DOS as punctuation or characters with special meanings, or for device names or reserved extensions.

Table 2-1 lists the characters which may be used in MS-DOS filenames and extensions.

Table 2-1. Valid Characters for Filenames and Extensions

A-Z	Alpha characters	()	Parentheses
a-z	Alpha characters	-	Hyphen
0-9	Numerics	_	Underscore
\$	Dollar sign	@	"At" sign
&	Ampersand	^	Carat
#	Pound sign	{ }	Braces
%	Percent sign	~	Tilde
'	Apostrophy	'	Single quotation
!	Exclamation point		



Table 2-2 lists the characters which may not be used in MS-DOS filenames and extensions.

Table 2-2. Invalid Characters for Filenames and Extensions

.	Period*	+	Plus sign
[]	Brackets	*	Asterisk**
?	Question mark**	:	Colon
\	Backslash	;	Semicolon
/	Forward slash		Space
=	Equal sign	< >	Angle brackets
"	Quote		Vertical Bar
,	Comma		

* A Period can only be used to separate a filename and extension. It may not be used within either.

** The Question Mark and Asterisk are interpreted as "wildcard characters" by MS-DOS (see Wildcards).

In addition to several reserved characters, MS-DOS has certain filenames and extensions which are predefined. All MS-DOS device names (listed in Table 2-4) are reserved and cannot be used as filenames. For example, the file name AUX is invalid. Appending an extension to an MS-DOS device name (i.e. AUX.BAS) will not make it valid.

In addition to device names, there are several extensions which have a predefined use in MS-DOS. These extensions should only be used for their specified purpose, but MS-DOS does not prevent you from using them for any purpose you choose. They are listed in Table 2-3.

Table 2-3. MS-DOS Predefined Extensions

Extension	Specified Use
.BAK	Backup file created by EDLIN (the MS-DOS editor) and several word processing programs.
.BAS	BASIC uses this extension for its program files.
.BAT	MS-DOS uses this extension for all Batch Command files.
.COM	MS-DOS program file.
.EXE	MS-DOS program file.
.\$\$	MS-DOS uses this extension for temporary files.

Note



You should check your application manuals for other predefined extensions which they may use.

MS-DOS Reserved File Names

There are certain filenames which have special meaning to MS-DOS. These are called device names. You may not name files with a device name. You may, however, use a device name in place of a filename in MS-DOS commands. The following is a list of reserved device names.

Table 2-4. MS-DOS Reserved Device Names

Name	Function
------	----------

CON	Console. This device name represents the keyboard for input and the screen for output. If used as an input filename in an MS-DOS command, press F6 , then Enter to signify the end of input.
AUX COM1	Serial Port 1.
COM2	Serial Port 2.
PRN LPT1	Parallel Port 1. This device supports output only.
LPT2	Parallel Port 2. This device supports output only.
LPT3	Parallel Port 3. This device supports output only.

Name	Function
NUL	Dummy device. When used as an input device, it immediately returns an End-Of-File condition. As an output device, it accepts data, but immediately discards it (i.e. no data is written).
CLOCK\$	This is a special purpose device used to read or set the system time and date. Unlike the other device names, CLOCK\$ may not be used in an MS-DOS command.

Wildcard Characters

The last topic pertaining to file naming is wildcard characters. These characters provide a means of referencing multiple files with one file name. Wildcards can be used with many MS-DOS commands. We will introduce Wildcards here in concept, and demonstrate their actual use later in this manual as we discuss the various MS-DOS commands.

There are two wildcard characters; the question mark "?" and the asterisk "*". The "?" wildcard means "substitute any single valid character". To illustrate the way the "?" wildcard works, assume the disc in Drive A: contains the following commands.

```
CHPT1.TXT    CHPT5.TXT
CHPT2.TXT    CHPT6.TXT
CHPT3.TXT    CHPT7.TXT
CHPT4.TXT    CHPT8.TXT
```

The file name CHPT?.TXT will reference them all. The digits 1 - 8 are substituted for the "?" wildcard character. An example of how this feature can be used would be to enter the MS-DOS command line

```
A>DIR CHPT?.TXT
```

DIR would list all eight of the files in the directory listing.

The "*" wildcard character operates in a similar manner. It is interpreted by MS-DOS to mean "fill the remaining characters in the filename or extension with the '?' wildcard". This wildcard can be used to reference all files with the same filename or same extension. For example, the files listed above could have been referenced with the command line

```
A>DIR *.TXT
```

In addition to the files listed above, any other files with an extension of .TXT would be included in the directory listing. Likewise, any files with a filename of BOOK would be included in the directory list produced by the following command line

```
A>DIR BOOK.*
```

The "*" wildcard is quite flexible. It can be used with fixed characters in either the filename or extension. The command line

```
A>DIR EXAMPLE.*
```

will list the files with all extensions beginning with the letter B (such as .BAK, .BAS, etc.).

One common use of this wildcard is the "global" file name *.*. This means "all files", and may be used with some MS-DOS commands. The command line

```
A>COPY *.* B:
```

will copy all files in the Current Directory on drive A: to the Current Directory on drive B:.



File Name Summary

Some guidelines to keep in mind when selecting a file name are listed below.

- | | |
|-------------|--|
| Consistency | File naming should be performed in a consistent manner. You should make use of extensions wherever possible to indicate the type of data contained in the file. For example, you could decide on an extension of .TXT for all text files. Consistency will also help you derive maximum benefit from the use of wildcards. |
| Clarity | Within the limits of eleven characters, the file name should be as descriptive of the contents as possible. If you are using a spreadsheet program to forecast Fall Sales, use the filename FALLSALE.FOR, instead of FORECAST. This will make choosing a file name for the Winter Sales forecast easier, and eliminate your wondering in the Summer which forecast is in the file FORECAST and which is in the file FORECST. |
| Conformity | The filename and extension must not contain any invalid characters or Device names. In addition it should not use any extensions reserved by MS-DOS for special uses. |

File Attributes

MS-DOS files can be assigned various attributes. Each file has four attributes. Each of the attributes may be enabled or disabled for the file. The four attributes are listed below.

Read-only This attribute marks a file as read-only. Files with this attribute set cannot be erased or updated. They can be renamed, however. The ATTRIB command modifies the read-only attribute.

Hidden This attribute marks a file as Hidden. Files with the Hidden attribute are normally excluded from directory searches by most application programs, and are always excluded by MS-DOS. Your MS-DOS System Disc contains three hidden files; the two MS-DOS files IO.SYS and MSDOS.SYS plus the PAM file PAM.CIF. An attempt to perform any MS-DOS file-oriented command on a hidden file will return an error message, such as

F i l e N o t F o u n d

System File This attribute is similar to the Hidden attribute. All files with this attribute are excluded from directory searches by most application programs.

Archive This attribute is used by BACKUP and RESTORE to perform incremental file back-up. This attribute is set by BACKUP and cleared by MS-DOS anytime the file is updated or extended.

Root Directories, Subdirectories, and File Allocation Tables

The task of keeping track of data on a disc is one of the most important performed by MS-DOS. A three-part system is used to keep track of the various files on a disc. This system is flexible enough that it can be used on a 64 Kbyte virtual disc (VDISK) and a hard disc.

MS-DOS uses Root Directories, Subdirectories, and a File Allocation Table to perform this task. Each of these components will be examined in this section.

Directories

Directories are used by MS-DOS to keep track of the files on a disc. Directories come in two types: Root Directories and Subdirectories. First we'll look at the characteristics which Root Directories and Subdirectories have in common, then we identify the differences between them.

Each Directory consists of a series of Directory Entries. Each file has a corresponding Directory Entry which contains pertinent information about the file.

Each Directory Entry contains the following pieces of information for each file: filename, extension, size of the file in bytes, date and time of creation or last update, file attributes, and a pointer to the file's first cluster. (We'll discuss clusters with the File Allocation Table). This information is displayed (with the exception of file attributes and starting cluster) by the DIR command.

Root Directory

Each disc has a single Root Directory. This root directory may be viewed as the cornerstone of the directory structure for the disc. The Root Directory has a predetermined number of Directory Entries, based on the type of disc media. The number of Directory Entries for each of the available disc types is shown in Table 2-5.

Table 2-5. Directory Entries in Root Directory

Drive	Capacity	Directory Entries
Flexible	160 Kbytes	66
Flexible	320/360 Kbytes	112
Flexible	1.2 Mbytes	224
Hard	20 Mbytes	512

Subdirectories

On discs with large capacities, a single directory may not be sufficient to adequately organize your data. MS-DOS provides the ability to add additional directories, called Subdirectories to your disc to solve this problem.

Subdirectories are very similar to the Root Directory we discussed earlier. They contain entries for each of the files in the directory and assist MS-DOS in locating and accessing files. However, unlike the Root Directory there is no limit to the number of directory entries which a Subdirectory can contain.

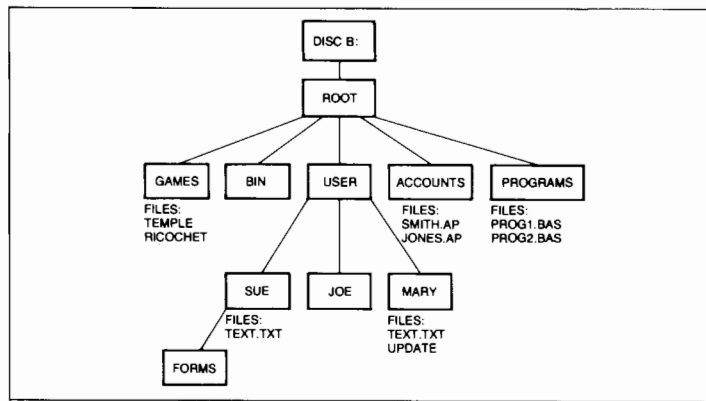


Figure 2-2. MS-DOS Directory Structure

As you can see from the figure, subdirectories are nested. This structure is also sometimes referred to as tree structured since it looks a bit like an inverted tree.

Subdirectories have levels which indicate how far away from the Root Directory they are. The first level (GAMES, BIN, etc. in Figure 2-2) is nested in the Root Directory. The second level is nested in the first, and so on.

Subdirectories use the same naming conventions as files; an eight character name and an optional three character extension, separated by a period. Subdirectories are created with the MKDIR command.

File Allocation Table (FAT)

When information is added to a file, disc space is allocated to the file in units of “clusters”. The size of a cluster is dependent on the size of the disc, but is generally 1Kb-8Kb. If a file uses all the space in a cluster, another cluster is allocated to the file. Because adjacent parts of a file are not necessarily stored in physically adjacent clusters, MS-DOS creates a File Allocation Table on each disc to locate all the clusters which comprise each file on the disc.

Because the integrity of the FAT is so important to the integrity of all the files on a disc, MS-DOS creates two copies of the FAT on each disc. One of the functions of CHKDSK is to verify the integrity of the FATS on a disc. If any problems are found, CHKDSK will report them and attempt to fix them. It is a good idea to run CHKDSK periodically to make sure the two FATS are intact.

Active Drive and Current Directory

Now that we've discussed directories and files, there are two terms which need defining; Active Drive and Current Directory. These terms will be used extensively throughout this manual.

Active Drive

MS-DOS assigns one of the disc drives as the Active Drive. In order to locate data or a program, MS-DOS must not only know the name of the file, but also the drive on which the file resides. A drive designator can be entered in front of a file name to explicitly tell MS-DOS where the file is located. The following tells MS-DOS that the file EXAMPLE.BAS is located on Drive B:

```
B:EXAMPLE.BAS
```

The Active Drive allows the drive designator to be implicitly referenced. If MS-DOS encounters a file name which isn't preceded by a drive designator in a command line, it will assume that the file is on the Active Drive. If the Active Drive is A:, MS-DOS would interpret

```
EXAMPLE.BAS
```

as

```
A:EXAMPLE.BAS
```


When MS-DOS starts it sets the Active Drive to the drive which it was booted from. For example, if you boot from the flexible disc drive, the Active Drive is A:. Booting from the hard disc will make C: the Active Drive.

The default MS-DOS system prompt (the system prompt can be changed with the MS-DOS PROMPT command) displays the Active Drive in the prompt. This will remind you of the Active Drive as you use your system.

The Active Drive may be changed by entering the drive designator of the new Active Drive at the system prompt. To change the Active Drive from A: to C:, enter the command line

```
A>C:
```

MS-DOS will respond with the new system prompt

```
C>
```

showing C: as the Active Drive.

Current Directory

The Current Directory is the directory (Root Directory or Subdirectory) which MS-DOS will assume is being referenced, unless a file name is preceded by a Path (see next section in this chapter).

Each drive has a Current Directory. When MS-DOS is booted, the Current Directory for each drive is the Root Directory. The Current Directory for a drive may be changed with the MS-DOS CHDIR command. We'll discuss CHDIR and give examples for its use in the next section.

MS-DOS Paths

A Path informs MS-DOS and application programs where to find a directory on a disc. A Path is literally the “path” which must be traversed to get to the subdirectory.

Let’s use a couple of examples from the directories shown in Figure 2-2 to illustrate Paths. The Root Directory has no name (unlike the subdirectories), and therefore is notated with the backslash character ‘\’. We’ll use the CHDIR command during these illustrations. CHDIR changes the current directory on a drive from one directory to another.

The command line

```
C>CHDIR \
```

will change the current directory to the Root Directory of Drive C:. The Root Directory character, ‘\’ may be preceded by an optional drive designator if you wish to reference the Root Directory on a drive other than the Active Drive. Thus the Root Directory on Drive A: would be notated A:\, Drive B: would be B:\, and so on.

Changing the current directory to the Subdirectory PROGRAMS could be accomplished with the command line

```
C>CHDIR \PROGRAMS
```

The Path to the Subdirectory PROGRAMS starts with the Root Directory symbol ‘\’, indicating that it is nested in the Root. To change the current directory to the Subdirectory SUE, the command line

```
C>CHDIR \USER\SUE
```

could be entered. Notice that the backslash character has a dual purpose; it indicates Root Directory, and it serves as a punctuation mark separating subdirectory names.

Note in the above example that if the current directory was \USER, then

```
C>CHDIR SUE
```

would have produced the same result. If a path name starts with a backslash, then the path starts at the root. If not, the path starts at the current directory.

Files and Paths

So far we have discussed files and directories separately. It is now time to look at them together. Each file has its own unique "address" or "ID" which consists of the directory the file resides in and its file name. This is often referred to as a File Specifier.

Often it will be necessary to refer to a file which is not in the current directory. To do so requires use of the file's Pathname. A Pathname consists of the Path, identifying the directory, and a File Name identifying the file. The Path and File Name are separated by a backslash as in the following example.

```
C:\SUB1\SUB2\SUB3\FILENAME.EXT
```

This references the file FILENAME.EXT in the subdirectory "\SUB1\SUB2\SUB3" on drive C:.

Note

Throughout the remainder of the manual, frequent references will be made to directories. The term Pathname will be used to describe a Path and File Name. There is a difference between Path and Pathname.

There is no limit to the depth which subdirectories can be nested, but the Pathname must not be longer than 63 characters. Note that this limit is for the entire Pathname from the Root Directory, even if part of the pathname is implicitly rather than explicitly referenced! Any attempt to reference file or subdirectory using a Pathname in excess of 63 characters will produce the error message

`Invalid Directory`

Part of a Pathname may be implicitly referenced. If the Path of the Current Directory is

`C:\SUB1\SUB2`

the Pathname of the file EXAMPLE.BAS is

`C:\SUB1\SUB2\EXAMPLE.BAS`

which is a total of 24 characters. However, since C:\SUB1\SUB2 is the Current Directory, we can reference the file simply as

`EXAMPLE.BAS`

which is only 11 characters. The Path (of the subdirectory) is implicitly referenced. However, any implicitly referenced Path must be counted in the total length of the PATH.

Let's take a look at a few valid Paths and Pathnames.

\	Root Directory, active drive.
C:\	Root Directory, Drive C:.
\SUB1\FILENAME	File FILENAME, Subdirectory SUB1, active drive.
A:\SUB1\FILENAME.EXT	File FILENAME.EXT, Subdirectory SUB1, drive A:.
\SUB1\SUB2\SUB3	Subdirectory SUB3, active drive.
B:\SUB1\SUB2\SUB3	Subdirectory SUB3, drive B:.
\S\S2\FILENAM	File FILENAM, Subdirectory SB2, active drive
D:\SUB1\SUB2\FILENAM	File FILENAM, Subdirectory SUB2, drive D:.
FILENAME	File FILENAME, current directory, active drive.
C:FILENAME.EXT	File FILENAME.EXT, current directory, drive C:.
SUB1\FILENAME.EXT	File FILENAME.EXT, subdirectory SUB1 of the current directory.

3

MS-DOS Command Basics

What is an MS-DOS Command?	3-1
Internal Commands	3-1
External Commands	3-1
The MS-DOS Command Line	3-2
The MS-DOS System Prompt	3-2
Constructing A Command Line	3-3
Executing A Command Line	3-5
MS-DOS Commands	3-5
File-Oriented Commands	3-6
Disc-Oriented Commands	3-8
Peripheral Commands	3-10
System Commands	3-11

3

MS-DOS Command Basics

In this chapter, we examine the basics of MS-DOS commands. We'll discuss the types of MS-DOS commands, how to enter a command line, and list the MS-DOS commands.

What is an MS-DOS Command?

There are two types of MS-DOS commands; internal and external. Internal commands reside in memory, and are always available for use. External commands, on the other hand, are stored on disc as files. External commands are really no different than your application programs except that they manage your files and system rather than process words, create spreadsheets, etc.

Internal Commands

The MS-DOS internal commands are part of the larger program COMMAND.COM. They are available any time the MS-DOS system prompt is displayed on the screen.

External Commands

External commands operate the same as internal commands, with the exception that they must first be loaded into memory from disc. In order to execute an external MS-DOS command, it must be on one of the discs in the system. External MS-DOS commands will have one of three possible file extensions, .COM, .EXE, or .BAT.

The .BAT extension is a special type of MS-DOS command. A file with this extension is called a Batch Command File and is processed with the MS-DOS Batch processing capability. Batch commands are the topic of the chapter titled *Batch Processing*.

The MS-DOS commands are listed at the end of this chapter and described in detail in the chapter titled *MS-DOS Command Descriptions*.

The MS-DOS Command Line

MS-DOS commands are entered via a command line. In this section we'll examine the MS-DOS command line in detail.

The MS-DOS System Prompt

MS-DOS commands can only be entered when the MS-DOS system prompt is displayed on the screen. The standard MS-DOS system prompt consists of the active drive designator followed by the > (greater than) character. If the active drive is A:, the system prompt will be

```
A>
```

MS-DOS has the capability of displaying the system prompt in different formats; the example above is only the default. The PROMPT command may be used to modify the system prompt to include the time, date, current directory, MS-DOS version number, or any characters you wish. For example, the PROMPT command could produce the following system prompt

```
A>:\SUB1>
```

indicating the current directory.

Refer to the chapter titled *MS-DOS Command Descriptions* for specifics on use of the PROMPT command. Throughout this manual, examples will be displayed using the default system prompt.

Constructing A Command Line

An MS-DOS command line consists of several distinct parts. To illustrate the components in a representative command line, we'll use the FORMAT command, as shown in Figure 3-1.

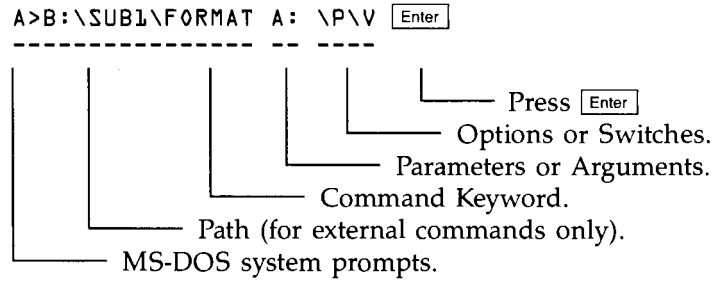


Figure 3-1. MS-DOS Command Line

Let's briefly review each of the components. As stated earlier, the MS-DOS system prompt must be displayed to indicate that MS-DOS is ready to accept and execute a command line.



Next is the optional Path. This may be included to instruct MS-DOS which directory to look in for the command file. In the example above, the file FORMAT.COM is located in the subdirectory \SUB1 on drive B:. MS-DOS will look in that directory. The path is not allowed with internal commands. If it is used the error message

```
Bad command or file name
```

will be displayed.

If no Path is included in a command line for an external command, MS-DOS will search the accessible directories to find it. Accessible directories are the Current Directory, plus any additional directories specified with the MS-DOS PATH command. For more information, refer to the PATH command in the chapter titled *MS-DOS Command Descriptions*.

The command Keyword follows the optional Path. This is simply the name of the command, excluding the extension for external commands.

The last two parts tell the command what you want accomplished. The parameters or arguments depend on the command, but typically contain filename, drive designators, pathnames, etc. Many MS-DOS commands support options or switches. They consist of the forward slash character '/' and a letter.

A command line is terminated by pressing the key. This indicates to MS-DOS the command line is complete, and ready for execution.

The command line may be typed in upper or lower case characters, or a mix of both. MS-DOS converts all characters in the keyword to upper-case before executing it. Parameters and options are passed to the command exactly as entered.

Executing A Command Line

Many MS-DOS commands will prompt you during execution. Certain commands will ask for additional parameters. These should be typed in the format required for the command. Several commands have "safety checks" built into them. For example, if you enter the command line

```
DEL *.*
```

(erase all files from the current directory), DEL will make sure by asking

```
Are you sure? (Y/N)
```

If you enter Y or y, the command will continue execution. A N or n will terminate the command. The prompt will be reissued if any characters other Y or N are entered in response. This gives you the opportunity to double check potentially damaging commands before they are executed.

At times you may be given instructions to perform some task, then notify MS-DOS you've completed it. For example, FORMAT will instruct you to insert the disc you wish formatted. You are asked to press any key when ready to indicate you have completed the task. You should press the , or any of the alphanumeric keys (A-Z, 0-9).

MS-DOS Commands

In this section the MS-DOS commands are briefly outlined. They are broken down into four different groups, based on their function. These groups are; File-Oriented commands, Disc-Oriented commands, Peripheral Commands, and System Commands. Each command is followed by a brief description of its function. A detailed description of each command appears in the chapter titled *MS-DOS Command Descriptions*.

**File-Oriented
Commands**

Name	Description
ATTRIB	Changes a file's Read-only attribute.
BACKUP	Backs up files from one disc to another.
COMP	Compares two files.
COPY	Copies a file.
DEL	Removes a file from a disc.
DIR	Performs a directory listing.
ERASE	Removes a file from a disc.
EXE2BIN	Converts a file from .EXE to binary object file format.
FC	Compares two files.

Name	Description
PATH	Sets alternate directory searches for a command or program.
RECOVER	Recovers damaged files or directories from a disc.
RENAME	Renames a file.
RESTORE	Restores files to a disc which were originally backed up with the BACKUP command.
SHARE	Loads MS-DOS support for shared files in a network environment.
TREE	Lists all subdirectories on a disc by Path.
TYPE	Displays the contents of a file on the Standard Output device.

**Disc-Oriented
Commands**

Name	Description
ASSIGN	Assigns one logical disc drive to another.
CHDIR	Changes the Current Directory for a disc.
CHKDSK	Checks the validity of a disc, and reports disc and system memory statistics.
DISKCOMP	Compares the contents of two different discs.
DISKCOPY	Copies the contents of one disc to another.
FDISK	Manages MS-DOS Partitions on a hard disc.
FORMAT	Formats a disc for use by MS-DOS.
LABEL	Places a volume label on a disc, or changes an existing volume label.
MKDIR	Creates a Subdirectory.
RMDIR	Removes a Subdirectory.

Name	Description
SUBST	Assigns a Subdirectory to a drive designator.
SYS	Transfers a copy of MS-DOS and optionally PAM to a disc.
VOL	Displays the volume label of a disc.

Peripheral Commands

Name	Description
CTTY	Changes the Physical device assigned to the Standard Input and Output devices.
GRAFTABL	Loads an alternate character set for the display adapter.
GRAPHICS	Loads support for printing of display screens in the Graphics mode on the system printer.
KEYBUS	Provides support for foreign language keyboards.
MODE	Sets operating parameters for the system parallel and serial ports, and controls Standard Printer assignment.
PRINT	Prints the contents of a file(s) as a background task

System Commands

Name	Description
BREAK	Enables or disables extended CTRL-Break and CTRL-C checking by MS-DOS.
CLS	Clears the display screen.
DATE	Displays and/or sets the current system date.
PROMPT	Specifies the format of the system prompt.
SET	Sets or displays parameters in the MS-DOS system environment.
TIME	Displays and/or sets the current system time.
VER	Returns the current version number of MS-DOS.
VERIFY	Enables or disables disc write verification by MS-DOS.

4

Batch Processing

What is Batch Processing	4-1
Creating a Batch Command Line	4-2
Executing a Batch Command File	4-3
Chaining Batch Command Files	4-4
AUTOEXEC Batch File	4-5
Batch Command Files With Replaceable Paramaters ..	4-6
Using %0 - %9	4-6
Using the SET Command	4-8
Batch Commands	4-9
ECHO	4-10
FOR	4-12
GOTO	4-14
IF	4-16
PAUSE	4-18
REM	4-20
SHIFT	4-21

4

Batch Processing

In this Chapter we'll examine a powerful feature of MS-DOS, Batch Processing. As you use MS-DOS you may find yourself executing certain sequences of commands frequently. These command sequences can be automated using the MS-DOS Batch Processing capability.

What is Batch Processing

Batch Command Processing allows multiple MS-DOS commands and application programs to be placed in a file, or batch, and executed as a single command. The Batch Processor in MS-DOS will execute the commands which were previously entered into the file, saving you from entering them yourself.

Batch processing also has its own set of special commands which provide a rudimentary "program language", further enhancing the capabilities of MS-DOS Batch Processing. We'll discuss them later in this chapter.

There is one limitation to using Batch Processing which you should be aware of. The response to any command prompt must still be typed at the keyboard, and cannot be included in the Batch file. This prevents using the Batch command for unattended system operation with MS-DOS commands which issue prompts. Even so, Batch command files which include these commands are still useful for eliminating keyboard entry of command lines.

Creating a Batch Command Line

There are numerous ways to create a Batch command file. You can use a word processor or editor which stores text in ASCII format; EDLIN, the MS-DOS editor; or the COPY command. We'll use the COPY command in our examples, since it is usually the easiest way to create a Batch command file.

A Batch command file must have an extension of .BAT. The filename can be any valid MS-DOS filename. The COPY command can be used to create a file using the keyboard. If the source parameter in the command line is the Console Device, CON, instead of a file, COPY will enter characters typed at the keyboard into the destination file until an end-of-file character is entered. The end-of-file character **CTRL-Z** can be entered by pressing the **CTRL** and letter Z keys simultaneously, or by pressing function key **F6**.

The filename cannot be the same as any of the MS-DOS commands, internal or external. MS-DOS will always execute a command or program with an extension of .COM or .EXE over a Batch command file with that filename. MS-DOS will allow you to create the Batch command file, but you'll be unable to execute it.

Let's create a sample Batch Command file with the MS-DOS commands DIR and CHKDSK in it. We'll name the file CHKDIR.BAT, and use COPY to create it.

```
A>COPY CON CHKDIR.BAT
DIR
CHKDSK
CTRL Z
1 File(s) copied

A>
```

Executing a Batch Command File

Executing a Batch command file is very similar to executing any other MS-DOS command. The filename is simply entered at the MS-DOS system prompt without the .BAT extension. To execute the Batch command file we just created, enter the command line

```
A>CHKDIR
```

This Batch command file will list the directory of the current directory on drive A:, then execute CHKDSK on A: to determine the amount of disc space available.

```
A>DIR
```

```
Volume in drive A is SYSTEM
Directory of A:\

COMMAND  COM   23210  3-17-85  12:00a
ANSI     SYS   2501  3-12-85  10:28p
VDISK    SYS   2626  4-09-85  4:55p
.        .      .      .        .
.        .      .      .        .
.        .      .      .        .
CONFIG   SYS    128  6-11-85  8:49a
          21 File(s)  153600 bytes free
```

```
A>CHKDSK
```

```
Volume SYSTEM created May 23, 1985 3:38p
```

```
362496 bytes total disk space
 38912 bytes in 3 hidden files
169984 bytes in 21 user files
153600 bytes available on disk
```

```
655360 bytes total memory
215744 bytes free
```


You can terminate Batch command file execution using either of the MS-DOS abort commands, **CTRL-Break** or **CTRL-C**. However, you will not return directly to the MS-DOS command prompt. Instead, the prompt

```
Terminate batch job (Y/N)?
```

will appear on the screen. If you enter Y or y you'll be returned to the MS-DOS system prompt. Otherwise, Batch processing is resumed with the next command line in the Batch command file. The command line that was executing when the **CTRL-Break** or **CTRL-C** was entered is not resumed.

It is permissible to change the current directory and/or active drive during the execution of a Batch command file. MS-DOS "remembers" the drive and directory where the Batch file is. However, the disc with the Batch command file can not be removed from the drive. If it is, MS-DOS will issue the error message

```
Not ready error reading drive <d>:  
Abort, Retry, Ignore?
```

In order to continue execution of the Batch command file, the disc with the command file must be re-inserted into the original drive, and R for Retry entered. Entering A will terminate the Batch command file and return you to the MS-DOS system prompt; I is not recommended.

Chaining Batch Command Files

Batch command files can be "chained" for execution, that is one Batch command file can contain another Batch command file as a command line. This allows a long string of command lines to be assembled out of several smaller Batch command lines. There is no limit to the number of Batch Command files which can be chained together.

The command line containing the filename of the Batch command file to chain to should be the last one in the file, since all command lines after that command line will not be executed. Once a Batch command file has invoked a second Batch command file, there is no way to return to the first (except to chain to it).

AUTOEXEC Batch File

MS-DOS has a special Batch command file called the AUTOEXEC Batch file. It is a regular batch file with the name AUTOEXEC.BAT which is placed in the Root Directory of the boot drive. During the boot process, MS-DOS will look for this file in the directory. If AUTOEXEC.BAT is found, MS-DOS will execute it prior to displaying the system prompt.

An AUTOEXEC.BAT file is very useful for executing MS-DOS commands which initialize or configure the system in some way. For example, if you want the extended `CTRL-Break` checking to be active all the time, the command line

```
BREAK ON
```

could be placed in an AUTOEXEC.BAT file. MS-DOS would then execute this command each time it was booted.



Batch Command Files With Replaceable Parameters

MS-DOS allows you to create Batch command files with replaceable parameters. This allows you to create the Batch command files without specifying all parameters. There are two ways to replace parameters:

- The %0-%9 positional parameters
- The SET Command

Using %0-%9

The ten positional parameters %0-%9 can be inserted into Batch command files to specify a replaceable parameter. The replacement parameters are entered on the command line when the Batch command file is executed. The first parameter in the command line replaces %1, the second %2, and so on. %0 is always replaced by the file specification of the Batch Command file entered in the command line. Thus, it does not include the .BAT extension. The %0 parameter is used in the following example, and should clarify this point.

To illustrate the use of replaceable parameters, let's create the following Batch command file.

```
C>COPY CON REPLACE.BAT
TYPE %0.BAT
DIR %1
CHKDSK %2
COPY %1*.* %2
CTRL Z
    1 File(s) copied

C>
```

When we execute it, we must include two parameters in the command line which will be substituted for %1 and %2. If we enter the command line

```
C>REPLACE A: B:
```

%1 will be replaced with A: and %2 with B:. The command lines which will be executed are

```
C>TYPE REPLACE.BAT
C>DIR A:
C>CHKDSK B:
C>COPY A:*. * B:
```

The Batch command file itself isn't modified.

If the % character is contained in a name in the Batch command file, it must be entered as a double % (%%) to differentiate it from the replaceable parameter character. To specify the file HAL%.EXE in the Batch command file, enter HAL%%.EXE.

If more parameters are encountered in the Batch Command file than there are replacement parameters in the command line, the unspecified parameters will be deleted. This may or may not cause problems as the Batch Command file executes. For example, if the example above had been invoked with the command line

```
C>REPLACE A:
```

(the second parameter is omitted) the Batch Command file would execute the following MS-DOS command lines.

```
C>TYPE REPLACE.BAT
C>DIR A:
C>CHKDSK
C>COPY A:*. *
```

In this instance the drive designator B: has been omitted, so this Batch Command file will execute differently than the previous example (i.e. CHKDSK will check drive C: instead of drive B:, and all files in the current directory of A: will be copied to C:).

The number of variables can be effectively increased from 10 with the use of the Batch command SHIFT. This command is discussed later in this chapter.

Using the SET Command

Another method of replacing parameters is to use the MS-DOS SET command. The SET command can place any number of name and parameter pairs in the System Environment (refer to the chapter titled *System Configuration* for additional details). The name is enclosed in the percent sign character (%), and inserted into the Batch command file. When the Batch command file is executed, the Batch command processor will search the System Environment for any names it encounters. The corresponding parameter will be substituted.

Let's start by assigning the parameter EXAMPLE.TXT to the name FILE. The following command line will accomplish this.

```
A>SET FILE=EXAMPLE.TXT
```

The name FILE is inserted in the Batch command file in the example shown below.

```
C>COPY CON LIST.BAT
TYPE %FILE%
CTRL Z
1 File(s) copied
C>
```

When this Batch command file is executed, the file EXAMPLE.TXT will be TYPed. The MS-DOS Environment is lost when the system is rebooted. Therefore, the SET command must be used to assign any names contained in Batch command files prior to their execution.

Batch Commands

This section discusses the Batch Commands. These Commands can be included in Batch command files along with regular MS-DOS commands and application programs to expand the capabilities of the Batch Command Processor. These commands should only be used within Batch command files.

ECHO Purpose

The ECHO command turns the screen display on or off during batch commands, or displays a message.

Syntax

```
ECHO [ON|OFF|message>]
```

Operation

ECHO ON instructs the Batch Command Processor to display all command lines as they are executed. This is the default state. Often, however, it is desirable to inhibit screen output.

ECHO OFF will suppress the display of command lines as they are executed. However, the display from the commands and application programs themselves will be displayed.

ECHO followed by a message will display the message regardless of the current ECHO state (On or Off). This form of the command functions identically to the REM command, with the exception that the message is displayed even in the ECHO OFF state.

If the ECHO command is entered with no additional parameters, the current state (ECHO ON or ECHO OFF) is displayed.

The following Batch command file utilizes all four forms of ECHO.

```
ECHO OFF
ECHO DRIVE B:
DIR B:/W
ECHO ON
ECHO
DIR A:/W
```



When this file is executed, the following display occurs:

```
A>ECHO OFF
DRIVE B:

Volume on drive B has no label
Directory of B:\

FILE1.TXT   FILE2.TXT

2 file(s)   265234 bytes free

A>ECHO
ECHO is on
A>DIR A:/W

Volume on drive A has no label
Directory of A:\

FILE1.TXT   FILE2.TXT

2 file(s)   265234 bytes free

A>
```


FOR Purpose

The FOR command allows iterative execution of MS-DOS commands.

Syntax

```
FOR %%<variable> IN (<set>) DO <command>
```

Operation

The FOR command accepts three arguments; variable, set, and command. The variable is a positional variable consisting of any single character. (It is best to avoid the digits 0-9 and hence any conflict with the positional arguments.) The variable is made equal to each value in the set. The set consists of a series of values (such as file names) separated by spaces, or a file name with one or more Wildcard characters. Below are two examples of valid sets.

```
(CHPT1.TXT CHPT2.TXT INDEX.TXT)  
(*.*TXT)
```

Each time the variable is made equal to a value in the set, the command line after DO is executed.

In most cases the variable will be included in the command line. The command is any DOS command.

The following command line in a batch file will find all of the chapter files for a book, and run them sequentially through an index program.

```
FOR %%A IN (CHPT?.TXT) DO INDEX %%A
```

%%A is set to the file name of each chapter (CHPT1.TXT, CHPT2.TXT, etc.) and the command lines INDEX CHPT1.TXT, INDEX CHPT2.TXT, etc., are executed.

Notes

1. The FOR command may not be nested. Only one FOR command can be included in each command line.
2. Path Names are allowed in the set of variables.
3. The FOR command may be run from the MS-DOS command prompt. The only difference is that the variable need only be preceded by one "%".

GOTO Purpose

The GOTO command transfers control in a batch file to the line following the line identified by a label.

Syntax

```
GOTO <label>
```

Operation

The label can be any string of characters. However, only the first eight characters are significant (used to differentiate it from other labels). A label must be preceded by a colon (:) and there should be no other commands in the line. The GOTO command will execute the command on the line following the label. The following example will transfer control to the command following the label :TWO

```
GOTO TWO  
.  
.  
.  
:TWO  
REM LABEL TWO
```

This example will print

```
REM LABEL TWO
```

Notes

- 1.** The GOTO command can be used with the IF command (see below). This provides IF-THEN branching in Batch Command files.
- 2.** Labels within a batch file are not displayed when the batch file is executed. Anything on the same line as

the label is ignored. Thus, a label which is not referenced may be used as a comment line and will not be displayed when the batch file is executed.

IF Purpose

The IF command allows conditional execution of MS-DOS commands in Batch Command files.

Syntax

```
IF [NOT] <condition> <command>
```

Operation

The IF command examines the <condition>, and executes the <command> if the condition is true, or if the <condition> is false and NOT is included in the command line. The following is a list of conditions which IF can test for.

ERRORLEVEL <number> This condition is true if the previous program executed had an exit code of <number> or higher. An exit code is set by a specific program. It is returned to the operating system when the program has finished.

<string1>==<string2> This condition is true if the two strings are identical. Either or both strings may be replaceable parameters. For example both of the following can be included in a Batch Command file.

```
IF %1==EXAMPLE
```

```
IF %1==%2
```

EXIST [<d>:] This condition is true if
[<path>]<filename>[.<ext>] the file specified exists.
The file name may include
the wildcard characters.

Note

1. BACKUP and RESTORE are the only DOS commands that return an ERRORLEVEL. This facility is provided for programs that may return an ERRORLEVEL which can be tested by this command.

PAUSE Purpose

The PAUSE command suspends execution of a batch file and allows you to display a message or an instruction to be performed before resuming.

Syntax

```
PAUSE [<message>]
```

Operation

The PAUSE command will temporarily suspend execution of a Batch Command file. An optional message of up to 121 characters may be displayed. This will be followed by the prompt

```
Strike a key when ready...
```

Pressing any key except **CTRL**-**Break** or **CTRL**-**C** will resume execution of the Batch Command file. Pressing either of these key sequences will terminate the Batch Command file.

The PAUSE command is useful for issuing instructions to the Batch Command user, and allowing sufficient time to execute those instructions. For example, the PAUSE command

```
PAUSE Insert disc in drive A:
```

will display the message and prompt

```
PAUSE Insert disc in drive A:  
Strike a key when ready...
```

This will instruct the user to place a disc in drive A:, and wait until they have pressed a key to indicate that the task is complete.

Notes

1. If a `CTRL-Break` or `CTRL-C` is entered, the prompt
`Terminate batch job (Y/N)?`
will be displayed. If you answer Y or y, you'll be returned to the MS-DOS system prompt. Answering N or n will return to executing the Batch Command file. The prompt will be repeated if any other characters are entered.
2. If ECHO is OFF, the optional message will not be displayed. However, the `Strike a key when ready` prompt will be.

REM Purpose

The REM command displays the message entered after the command word.

Syntax

```
REM [<message>]
```

Operation

The message entered with the REM command is displayed when the Batch Command Processor reaches that line. The message can be any string up to 123 characters in length. If the ECHO is OFF, the message is not displayed.

The following command line

```
REM Copy successfully completed.
```

will display

```
REM Copy successfully completed.
```

when the Batch Command file is executed.

SHIFT Purpose

The SHIFT command allows access to more than 10 replaceable parameters.

Syntax

```
SHIFT
```

Operation

Each time the SHIFT command is executed, the values assigned the replaceable parameters are shifted; %1 becomes %0, %2 becomes %1, and so on. The tenth parameter entered in the command line when the Batch Command file was invoked is assigned to %9.

With careful planning and use of the SHIFT command, the limit of 10 replaceable parameters can effectively be removed. Note however, that even with the SHIFT command only 10 parameters can be active at any one time. In addition, there is no way to retrieve parameters which have been "shifted out" (i.e. once a SHIFT is executed, the parameter %0 that existed before the shift cannot be recovered). The following example demonstrates the way SHIFT works. The values assigned each replaceable parameter are shown below:

```
%0 = Alpha  
%1 = Beta  
%2 = Gamma  
%3 = Delta
```

After the SHIFT command the values will be as follows:

%0 = Beta
%1 = Gamma
%2 = Delta

Note

1. The Alpha is lost, %3 is set to the previous value of %4, and if more than 9 parameters were entered in the command line, %9 is set to the 10th one.

5

Using the MS-DOS Keyboard








Editing a Command Line	83
Printing the Screen	84
Other Useful Functions	85
Unused Keys	86

5

Using the MS-DOS Keyboard

The MS-DOS keyboard has many features which are designed to make Vectra easier and faster to use. These features allow you to edit an MS-DOS command line, print a copy of your screen, cancel an MS-DOS command, and perform other useful functions. This chapter explains how to use these features.

Editing a Command Line

MS-DOS provides a set of editing commands which can assist in entering command lines. The MS-DOS editing keys are , , , , , , and .

The heart of this editing capability is the Template and the Template Pointer. Each time a command line is entered, MS-DOS stores a copy of it in the Template, and sets the Template Pointer to point to the first character in the Template. Characters can then be easily copied from the Template Pointer to create a new command line. For example, when the command line

```
C>COPY *.TXT B:
```

is entered, the Template will be set to

```
  COPY *.TXT B:
```

The template may now be edited to create the next command line. The editing keys and their respective

functions are listed below.

F1 or **▶**

Copies the character pointed to by the Template pointer to the command line each time it is pressed. The Template Pointer is then advanced one character in the Template. For example, if the Template contains COPY * .TXT B: and the current command line is

A>

after **F1** is pressed twice the command line will be

A>C0

F2

Copies characters from the Template to the current command line, starting at the Template Pointer, until a specified character is found in the Template. If the Template is COPY * .TXT B: and you type

F2 *

COPY will be added to the current command line.

F3

Moves the contents of the Template to the current command line, starting at the Template Pointer, until the last character in the Template is reached. If the current Template is COPY * .TXT B: and the current command line is

C>

after **F3** is pressed the command line

will be

```
C>COPY *.TXT B:
```

F4

Moves the Template Pointer forward to point to the next occurrence of the specified character. If the character is not found, the Template Pointer is not moved. This makes it easy to skip over large sections of a Template. For example, if the template is COPY *.TXT B: and you press

```
F4 *
```

the Template Pointer will point to the * .

F5

Moves the contents of the current command line to the Template. The previous contents of the Template are lost. If the Template is COPY *.TXT B: and the current command line is

```
A>FORMAT A: _
```

then **F5** will change the Template to FORMAT A:.

Ins

Normally, the Template Pointer is advanced when alphanumeric keys are typed on the command line. When **Ins** is pressed, subsequent alphanumeric keys will not cause the Template Pointer to be advanced, until **Ins** is pressed again. When used with the **▶** key, the **Ins** keys allow characters to be "Inserted" into the Template.

DEL Each time the **DEL** key is pressed, the character pointed to by the Template Pointer is deleted from the Template. The character is not displayed on the screen, and the current command line is unaffected. If the current Template is COPY * .TXT B:, and the Template Pointer points to the *, then after the DEL key is pressed, the Template will be

COPY .TXT B:

← or **◀** Moves the cursor and the Template Pointer one character to the left. The character on the screen is deleted, but the Template is unchanged.

ESC The **ESC** key cancels the current command line. The Template is unchanged.

Printing the Screen

The **Prt Sc** key on the numeric keypad is the key for printing output from MS-DOS. Output for the printer is sent to Parallel Port 1 unless a MODE command has been issued to redirect that output. If you use Parallel Port 2 (LPT2:), Serial Port 1 (COM1:), or Serial Port 2 (COM2:) as a print output device, additional utilities will be required. See the chapter titled *MS-DOS Command Descriptions* for details on the MODE command.

Shift
Prt Sc Pressing this key combination is like taking a "snapshot" of the screen. If any MS-DOS commands are executing, they will be temporarily suspended while the current contents of the screen are printed.

Note

MS-DOS can print the screen any time the screen is in alphanumeric mode. However, support for printing the screen if it is in a graphics mode is not resident in MS-DOS. See the GRAPHICS command in the chapter titled *MS-DOS Command Descriptions* for more information on how to print screens in graphics modes.

CTRL

Prt Sc

Pressing this combination of keys turns on "Printer Echo" mode. Pressing it again will turn the mode off. When Printer Echo mode is on, each character will be sent to the printer when it appears on the screen.

Note

This function will only work for characters output to the standard input/output device. It will not work for characters which are output to the screen through BIOS calls or by directly accessing the video hardware. See the chapter titled *Standard Input and Output* for more information.

Other Useful Functions

CTRL

Break

This key combination is used to cancel commands before they are finished. It is also used like the **ESC** key to cancel current input lines before **Enter** is pressed.

Each time a character is input or output using the Standard I/O, Standard Printer, or Standard Auxiliary devices, MS-DOS checks if **CTRL** **Break** has been pressed, and aborts the command if it has. However, many commands are “disc intensive”, and do not often input or output characters. Thus it may be impossible to quickly terminate one of these commands. The BREAK command is used to extend checking for **CTRL** **Break** to disc input and output. See the chapter titled *MS-DOS Command Descriptions* for more information on the BREAK command.

CTRL
Num lock

Press this key combination to temporarily stop all processing by MS-DOS. This will allow you time to read data that might otherwise scroll by too fast, for example during a long directory listing. Press any key to resume processing by MS-DOS.

F6

This key is equivalent to pressing **CTRL** Z.

F7

This key is equivalent to pressing **CTRL** @.

CTRL
Alt +

Press this key combination to increase the volume of your keyboard click. You must use the + key on the numeric keypad.


CTRL
Alt -

Press this key combination to decrease the volume of your keyboard click. You must

5-6 Using the MS-DOS Keyboard

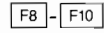
use the minus (-) key on the numeric keypad.

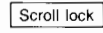


The  key moves the cursor to the next tab stop. Tab stops are set at every eighth character.

Unused Keys

The following keys are not used by MS-DOS. They may, however, be used by application programs that you run.



















Hard Disk Preparation

- Hard Disk Basics
- Partitioning
- FDISK
- Creating An MS-DOS Partition
- Changing The Root Partition
- Deleting An MS-DOS Partition
- Displaying Partition Information
- The FDISK Utility
- Creating A Primary Partition
- Creating A Logical Partition

6



Hard Disc Preparation

In this chapter we'll examine the steps necessary to prepare your hard disc for use. The FDISK command and disc partitions will be discussed.

Hard Disc Basics

The hard disc on your HP Vectra contains several rotating platters. These platters are comparable to a flexible disc; they are coated on two sides with magnetic recording material and each side is divided into tracks and sectors. The main difference is that each side of a platter contains over 600 tracks, compared to 40 on a standard capacity flexible disc, and 80 on a high capacity disc.

The platters are mounted on a spindle as shown in Figure 6-1. The top side of the upper platter is referred to as Side 0, the bottom side Side 1, the top side of the next platter Side 2, and so on. As you use the FDISK program, the term cylinder will be encountered. A cylinder consists of all similarly numbered tracks on all surfaces. For example, Cylinder 0 is comprised of Track 0—Side 0, Track 0—Side 1, Track 0—Side 2, etc. Thus each hard disc drive will have the same number of cylinders as tracks per side of each platter.

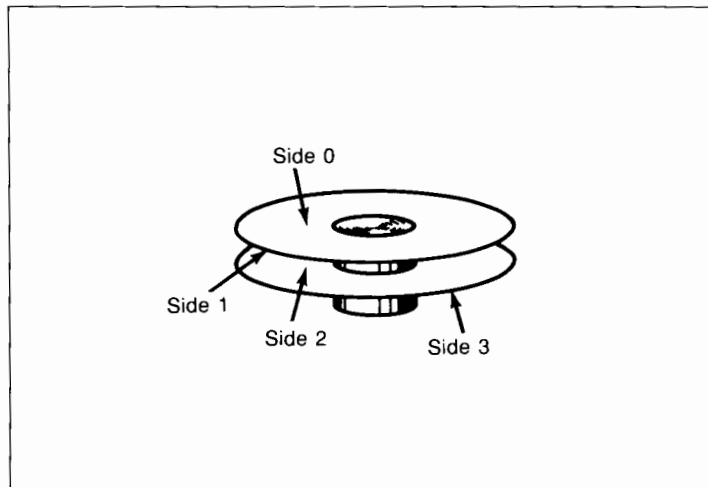


Figure 6-1. Hard Disc Drives

Partitioning

Hard discs may be subdivided into up to four partitions. Partitions allow different operating systems to reside on the same disc. Although operating systems perform the same basic function, they all have their own unique file structure, method of allocating disc space, etc. Therefore, they must have their own "disc". Partitioning allows the physical disc to be divided into smaller logical discs or partitions to facilitate the use of multiple operating systems.

These partitions may be any size, from one cylinder up to the total capacity of the disc. The size of each partition is specified in cylinders. Only one partition may be active at a time. An active partition simply means that the operating system may be booted from the hard disc drive. The disc is still accessible to the operating system if the partition isn't active, but MS-DOS will have to be booted from the flexible disc drive.

For example, if the MS-DOS partition on drive C: is active, MS-DOS can be booted from drive C: (provided of course, that MSDOS.SYS, IO.SYS, and COMMAND.COM are on drive C:). If the MS-DOS partition on drive C: isn't active, MS-DOS must be booted from drive A:. However, once MS-DOS is booted, all data contained in the MS-DOS partition on drive C: will be accessible.

The FDISK utility allows us to create a partition for MS-DOS, and to select the active partition. Although the MS-DOS partition created with FDISK might not comprise the entire disc, FDISK cannot create partitions for other operating systems. Each operating system used on the HP Vectra must have its own command to perform the task of creating a partition for that operating system.

FDISK FDISK operates slightly differently from other MS-DOS commands. It is menu-driven rather than command-driven. As such, there are no parameters or options to place in the command line.

The FDISK main menu is shown below.

```
FDISK Utility version 3.1
Fixed Disk Setup Program
Copyright (c) 1983-85 Phoenix
Software Associates, Ltd.
FDISK Options

Choose one of the following:

1. Create DOS Partition
2. Change Active Partition
3. Delete DOS Partition
4. Display Partition Data
5. Select Next Fixed Disk Drive

Enter choice: 1

Press ESC to return to DOS.
```

Option 5 (Select Next Fixed Disk Drive) will only be displayed if your HP Vectra is equipped with the optional 40 Mbyte hard disc drive. Selecting option 5 will allow you to create a partition on drive D:. Each of the other 4 options are discussed individually below.

Creating An MS-DOS Partition

The first step in creating an MS-DOS partition is to decide how large it should be. In most applications, MS-DOS will be the only operating system on the HP Vectra. Therefore, the MS-DOS partition should be the size of the disc. If you want to make more than one partition, decide how many cylinders you wish to allocate to the MS-DOS partition.

After selecting option 1 on the main menu, FDISK asks you if you want to use the entire fixed disc for MS-DOS with the following prompt:

Create DOS Partition

Current Fixed Disk Drive: 1

Do you wish to use the entire fixed disk
for DOS (Y/N)? Y

The default response is Y. Simply press the key if you want MS-DOS to use the entire disc. If you answered Y, FDISK will create the partition. Press to return to the FDISK main menu, and again to return to DOS. You will be prompted:

System will now reboot
Insert DOS diskette in drive A:
Press any key when ready...

Your fixed disc is now partitioned, and ready to be formatted using the MS-DOS FORMAT command. See the section titled *Formatting a Partition and Transferring MS-DOS and PAM* in this chapter.

If you do not wish to allocate the entire fixed disc to MS-DOS, enter N to the prompt

Do you wish to use the entire fixed
disk for DOS (Y/N)? N

FDISK will now display the prompt

Total fixed disk space is n cylinders
Maximum available space is n
Cylinders at n.

The first number is the total number of cylinders on the fixed disc drive. The maximum available space is the largest contiguous block of cylinders. If this is the first partition to be placed on the disc, then this number will be the same as the total number of cylinders. However, there may be other partitions already on the disc. Also, note that this is the largest contiguous block of cylinders, not the number of nonallocated cylinders.

FDISK will now display the prompt

```
Enter partition size.....:[n]
```

The number *n* displayed is the size of the largest block. If you wish to allocate the entire block, simply press the key. Otherwise, enter the number of cylinders you want to allocate to this partition.

Next the prompt

```
Enter starting cylinder number...:[n]
```

Again, the number *n* displayed is the beginning cylinder number of the largest block discovered by FDISK. Pressing the key will select that number, or you may enter one of your own choosing.

FDISK will now create this partition. If there is already a partition on the disc, chances are that it is the active partition. If you wish to boot MS-DOS from the hard disc, you'll need to change the active partition to the one just created using steps outlined in the next section. Once you make this partition the active partition, it will be ready to format using the MS-DOS FORMAT command.

Changing The Active Partition

In order to change the active partition, option 2 must be selected from the main menu of FDISK. The Change Active Partition menu shown below will now be displayed.

```
Change Active Partition
```

```
Current Fixed Disk Drive: 1
```

Partition	Status	Type	Start	End	Size
1	A	DOS	000	425	426
2	N	non-DOS	426	614	189

```
Total disk space is 615 cylinders
```

```
Enter the number of the partition you  
want to make active .....
```

There are two partitions listed in the menu above. Partition 1 is the current active partition. It is an MS-DOS partition, and occupies 426 cylinders on the disc. Partition 2 is currently inactive. It is a non-MS-DOS partition occupying the remaining 189 cylinders. In order to change the active partition from Partition 1 to Partition 2, enter the number 2 to answer the prompt. The non-MS-DOS partition is now active. If the hard disc drive is used as a boot drive, the operating system in Partition 2 will boot instead of MS-DOS.



Deleting An MS-DOS Partition

If you select option 3 in the FDISK main menu, you will enter the menu to delete an MS-DOS partition.

Caution



Deleting an MS-DOS partition will destroy all information stored in the MS-DOS partition. Be certain that all data and programs have been transferred from the MS-DOS partition to another system disc or backup discs.

The Delete DOS Partition menu is shown below.

```
Delete DOS Partition
```

```
Current Fixed Disk Drive: 1
```

Partition	Status	Type	Start	End	Size
1	A	DOS	000	426	426
2	N	non-DOS	426	614	189

```
Total disk space is 615 cylinders
```

```
Warning, data in the DOS partition  
will be lost. Do you wish to  
continue .....? N
```

Remember that the FDISK command cannot create or delete non-MS-DOS partitions. Thus, you will notice that FDISK does not ask you for the partition number, only if you want to delete the MS-DOS partition.

The default is N. If you made a mistake and don't want to delete the MS-DOS partition, press the key. Otherwise, type Y and press the key. The MS-DOS partition will be deleted.

Once the MS-DOS partition has been deleted, the hard disc drive will no longer be accessible under MS-DOS.

Displaying Partition Information

Option 4 of the main menu allows the current partition information for the hard disc to be displayed. The statistics for each partition are displayed in the following format.

```
Display Partition Information

Current Fixed Disk Drive: 1

Partition Status   Type   Start End Size
  1             A       DOS  000 425 426
  2             N     non-DOS 426 614 189

Total disk space is 615 cylinders
```

The partition number, status (active or non-active), type (MS-DOS or non-MS-DOS), starting cylinder, ending cylinder, and size (in cylinders) is listed for each partition. In addition, the total number of cylinders on the hard disc is displayed.

When you are finished examining the information, press the **[ESC]** key. You will return to the main FDISK menu.

Using FDISK With Drive D:

If your HP Vectra has the optional 40 Mbyte hard disc, then you have two hard disc drives, C: and D:. Although they are contained on the same physical disc drive, they are viewed by FDISK as if they were two separate drives. When FDISK is invoked, it checks to see if drive D: exists. If it does, option 5, Select Next Fixed Disc Drive, is displayed on the main menu. Unless option 5 is selected, FDISK assumes all references are to drive C:. If you select option 5, all subsequent FDISK operations will be performed on Drive D:. For drive D: then, select option 5 first, then any one of options 1 through 4.

Formatting a Partition and Transferring MS-DOS and PAM

Once you've created an MS-DOS partition and made it active, it must be formatted. Formatting a partition (hard disc) constructs the Root Directory and FATs, and initializes all sectors in the partition. The partition is formatted using the MS-DOS FORMAT command. It is recommended that you place MS-DOS and PAM on the hard disc, as well as give the hard disc a volume label. Insert a copy of the MS-DOS system disc in drive A: and enter the command line

```
A>FORMAT C:/P/V
```

If you don't wish to have PAM copied to the hard disc, enter the command line

```
A>FORMAT C:/S/V
```

This will only copy the MS-DOS files. You will still be prompted to enter a volume label since the /V option was specified.

Formatting the hard disc takes approximately 2 to 4 minutes. When the formatting process is complete, the MS-DOS and PAM files (if specified) will be copied to the hard disc. You will then see the following prompt asking you for the volume label.

```
Volume label (11 characters, Enter for none)?
```

Once you have entered the volume label, the formatting process is complete. Your hard disc may now serve as the boot disc.

You should now transfer the MS-DOS external commands and any application programs you want to the disc.

System Configuration

- Overview
- Configuration Commands
- BREAK
- BUFFERS
- COUNTRY
- DEVICE
- FCBS
- FILES
- LASTDRIVE
- SHELL
- Device Drivers
- DEVICE Command
- ANSI.SYS
- VDISK.SYS

7

System Configuration

In this chapter we'll examine the various ways in which MS-DOS can be expanded to suit the different system configurations and requirements of the HP Vectra.

Overview

MS-DOS is an extremely flexible operating system. It provides support for many different system components and configurations. This flexibility is the result of two different MS-DOS features.

The first is the set of MS-DOS commands which allow some of the parameters in MS-DOS to be altered to suit specific system requirements. These special MS-DOS configuration commands are placed in a system file, named CONFIG.SYS. CONFIG.SYS can be created using EDLIN or any word processor that creates unformatted files. Each time MS-DOS is booted, the configuration commands in CONFIG.SYS are executed. We'll examine these commands in the next section in this chapter.

The second is the ability to add extensions to MS-DOS using Device Drivers. Device Drivers allow peripheral devices not already supported by MS-DOS to be added to the system, and provide enhanced support for certain standard system components. We'll examine Device Drivers in greater detail later in this chapter.

Configuration Commands

There are eight MS-DOS configuration commands. Each operates in a manner similar to regular MS-DOS internal commands. Each command is followed by a set of parameters. The syntax is slightly different in that the command and parameters must be separated by an equal sign '='. With the exception of the BREAK command, configuration commands cannot be used outside of the CONFIG.SYS file. If they are entered in a standard MS-DOS command line, the error message

```
Bad command or file name
```

will be issued.

BREAK Purpose

The BREAK command enables or disables extended `CTRL-Break` checking by MS-DOS.

Syntax

`BREAK = ON | OFF`



Operation

MS-DOS checks for a `CTRL-Break` or `CTRL-C` keyboard sequence during all Standard I/O, Standard Printer, and Standard Auxiliary operations. If the extended `CTRL-Break` checking is enabled, MS-DOS will also check during disc operations.

The default state of the extended checking is off when MS-DOS is booted. Inserting the command

`BREAK=ON`

in the CONFIG.SYS file will enable the extended `CTRL-Break` checking.

Note

1. This command is functionally identical to the MS-DOS BREAK command.

BUFFERS

Purpose

The BUFFERS command specifies the number of disc I/O buffers.

Syntax

```
BUFFERS = <n>
```

Operation

n is the number of buffers, and may range from 1 to 99. The default number of buffers used by MS-DOS is 3. To increase the number of buffers to 10, place the command

```
BUFFERS = 10
```

in the CONFIG.SYS file.

The use of buffers may increase the speed of certain applications. As data is read from the disc, it is stored in a buffer. The buffers are used by MS-DOS in a manner that ensures that the most recently read data is in one of the buffers.

The performance of applications that perform a large number of random reads and writes on a data file may be increased by using additional buffers. As the number of buffers increases, the chances increase that the data requested by the application program is already in one of the buffers. Thus, MS-DOS can retrieve the data from memory instead of from disc, which speeds up program performance.

For applications which perform mostly sequential read and write operations, there is little performance gain from an increased number of buffers.

There is no precise formula to calculate the optimum number of buffers. Different values must be tried to determine the number of buffers that yields maximum performance. In general, applications which perform large amounts of random data access will perform best with between 10 and 25 buffers.

Notes

1. Each additional buffer specified increases the resident size of MS-DOS by 528 bytes. When a large number of buffers is specified, the amount of memory available to an application program may be significantly reduced.
2. In certain cases, a large number of buffers may actually decrease system performance. If the number of buffers is too large, it may take more time to search through the buffers than to read the data from disc.

COUNTRY

Purpose

The **COUNTRY** command is used to select the display format for the system time and date, currency symbol, and decimal separator based on the selected country.

Syntax

```
COUNTRY = <nnn>
```

Operation

nnn is a three digit country code. The corresponding code for each country is listed below.

Country	Code
United States	001
Netherlands	031
Belgium	032
France	033
Spain	034
Italy	039
Switzerland	041
United Kingdom	044
Denmark	045

Country	Code
Sweden	046
Norway	047
Germany	049
Mexico	052
Argentina	054
Venezuela	058
Austria	061
Finland	358
Israel	972

To select the United Kingdom display format for time, date, decimal separator, and currency symbol place the command

```
COUNTRY = 044
```

in the CONFIG.SYS file.

Notes

- 1.** The COUNTRY command does not translate MS-DOS messages and prompts, or change the video or keyboard character set.
- 2.** If your country is not listed above, pick a country code which most closely matches your needs.

DEVICE Purpose

The **DEVICE** command is used to load installable device drivers into the system.

Syntax

```
DEVICE = [<d>:][<path><filename>[.<ext>]
```

Operation

DEVICE specifies the file containing an installable device driver. MS-DOS will load the device driver from the file, and incorporate it into the resident portion of MS-DOS. Details about installable device drivers are contained in the next section of this chapter.

To install the MS-DOS virtual disc drive, **VDISK**, enter the command

```
DEVICE = VDISK.SYS
```

The size of the resident portion of MS-DOS will be increased by the size of each device driver installed.

Note

1. Two installable device drivers are provided with MS-DOS. They are **VDISK.SYS**, which provides the ability to convert system RAM into a virtual disc, and **ANSI.SYS**, which provides ANSI display terminal emulation for the Standard Output device.

FCBS Purpose

The FCBS command specifies the number of File Control Blocks (FCBs) which can be concurrently open with the Share Attribute set.

Syntax

```
FCBS= <x> , <y>
```

Operation

x sets the maximum number of files which can be concurrently open, and y the number of open files which are "protected" (see below). The command line

```
FCBS = 10 , 5
```

in the CONFIG.SYS file will allow 10 shared files open concurrently, with the first 5 "protected".

Some application programs use FCBs to keep track of files they are using. The x parameter specified in the FCBS command determines the maximum number of files which can be accessed with FCBs if the SHARE commands has been issued. The default value is 4 and the range is 1 to 255.

MS-DOS keeps track of which FCB (or file) was most recently used. If an application program attempts to open more files than specified and file-sharing has been loaded, MS-DOS will close the most recently used file prior to opening the requested file. The y parameter protects the first y files from being closed in this manner. If y is set equal to x, then files will not be closed and the application program will be unable to open any files in excess of the maximum. If the y parameter is not entered, MS-DOS will set it to 0.

Notes

1. The maximum number of files concurrently open applies only to shared files. No action is taken by MS-DOS if the maximum number of files is exceeded provided the SHARE command has not been executed.
2. If the FCBS command is included in the CONFIG.SYS file, the resident size of MS-DOS is increased.

FILES Purpose

The FILES command specifies the number of file handles which can be open concurrently.

Syntax

```
FILES= <n>
```

Operation

n is the number of file handles MS-DOS is to reserve space for. The default number is 8 and can range from 8 to 255. The command line

```
FILES = 10
```

in the CONFIG.SYS file will reserve enough room for 10 file handles.

Applications may create their own FCBs or they may open handles to access files. When an application opens a handle, MS-DOS creates a control block for the program. The FILES command determines the amount of space MS-DOS reserves for these control blocks, and hence the maximum number of handles which may be open concurrently.

Notes

1. Each handle over 8 increases the size of the resident size of MS-DOS 48 bytes.
2. There is no limit to the number of files an application program may have open when those files are opened through FCBs instead of handles, unless the SHARE command has been executed, in which case the maximum is set by the FCBS command.
3. This number is the total number of handles open for the entire system. A process may have a maximum of 20 files open at any time.

LASTDRIVE

Purpose

The LASTDRIVE command sets the maximum number of drives which may be accessed on the system.

Syntax

```
LASTDRIVE = <d>
```

Operation

d is any letter A through Z. The LASTDRIVE command sets the last valid drive designator for the system. The command line

```
LASTDRIVE = G
```

in the CONFIG.SYS file will provide 7 disc drive designators, A through G. The default value is 5 drives, A through E.

Each block device driver, subdirectory assigned via the SUBST command, and disc drive requires a drive designator. If the total of these exceeds five, a LASTDRIVE command must be placed in the CONFIG.SYS file to extend the last valid drive designator.

Note

1. If the last valid drive designator specified with the LASTDRIVE command is not high enough to accommodate the system disc drives and any installed block devices, the value specified in the command is ignored, and MS-DOS determines the highest required designator. In this situation though, there will be no unused drive designators for use with SUBST.

SHELL Purpose

The SHELL command allows an alternate command processor to be selected.

Syntax

```
SHELL = [<d>:][<path>]<filename>[.ext>]
```

Operation

The SHELL command substitutes an alternate command processor specified in the command line to replace COMMAND.COM. PAM is an example of an alternate command processor. It is substituted for COMMAND.COM using the following command line

```
SHELL = PAMCODE.COM ROOT
```

in the CONFIG.SYS file.

If this command is not included, MS-DOS starts the default command interpreter, COMMAND.COM.

Note

1. The word "ROOT" in the example above is interpreted by PAMCODE, and is not required for other Command Interpreters.

Device Drivers

Device Drivers allow additional peripheral devices to be added to MS-DOS. As we mentioned earlier in this chapter, Device Drivers provide the second major method for expanding MS-DOS.

MS-DOS accepts two types of Device Drivers, Character and Block devices. Block devices are disc drives, or devices which emulate them like "RAM discs", tape drives, bubble memory, etc. MS-DOS is supplied with one such Block device, the "RAM disc" driver VDISK.SYS (VDISK stands for "Virtual" DISK). We'll discuss how to install VDISK a little later in this section.

Character devices provide support to character-oriented peripherals such as modems, printers, plotters, etc. MS-DOS is supplied with a character device also, ANSI.SYS. ANSI.SYS provides emulation for ANSI standard terminal commands on the screen. Installing this device will also be discussed later.

DEVICE Command

The DEVICE command is used to install all Device Drivers, whether they are Character or Block devices. We've discussed the syntax of the DEVICE command already in this chapter. Now we will look a little deeper into how DEVICE installs the drivers. The specifics of the structure of Device Drivers are contained in the *MS-DOS Programmer's Reference Manual*.

Each driver has two characteristics which are important to DEVICE: device name and driver type. DEVICE first checks the driver type, since each of the two driver types are loaded somewhat differently.

The DEVICE command installs Character device drivers as they are encountered in the CONFIG.SYS file. The internal name of the device driver is checked against the default MS-DOS devices (see the chapter titled *Understanding MS-DOS Concepts*). If a driver with the same name is encountered, DEVICE loads it in place of the default MS-DOS device. This effectively allows the default MS-DOS character device drivers to be replaced by user-supplied drivers. For example, if the command line

```
DEVICE=LPT1.SYS
```

was encountered in the CONFIG.SYS file, and the driver had an internal name of COM1, that device driver would be loaded in place of the default system driver for the serial communication port. Device drivers which do not replace default drivers are loaded as system extensions. The single exception is the NUL device, which cannot be substituted.

When character device drivers are added, their internal device names are added to the list of reserved MS-DOS device names, and cannot be used as filenames.

When MS-DOS boots, it checks to see which drives are already installed in the system. The internal drivers for these drives are loaded, and drive designators assigned to them.

When the DEVICE command installs a block device, it is assigned the next disc drive designator. For example, when VDISK is installed in a system without the optional hard disc drive, the first VDISK will be assigned as drive C:. In a system with a hard disc, it would be drive D:. If more than one block device is specified in the CONFIG.SYS file, drive designators are assigned in the order the drivers are encountered in the file. Unlike Character device drivers, block device drivers cannot replace internal system drivers. In addition, the names of installed block device drivers (i.e. VDISK, etc.) are not added to the list of reserved MS-DOS device names.

ANSI.SYS

The ANSI.SYS driver supports extended keyboard and screen display features. This driver is installed by inserting the command line

```
DEVICE=ANSI.SYS
```

in the CONFIG.SYS file. When this driver is installed, ANSI Standard Terminal Escape sequences will be executed by the Standard Input and Output device. ANSI.SYS replaces the CON device driver. The sequences are described in the appendix titled *Extended Screen and Keyboard Control*.

VDISK.SYS

The VDISK block device driver allows a portion of the HP Vectra's RAM memory to be used as a disc drive. These block devices (which emulate a physical disc drive) are sometimes referred to as "RAM Discs" or "Virtual Discs". The Virtual Discs can be used in the same manner as other system disc drives with the exception of a few MS-DOS commands such as FORMAT, CHKDSK, etc. VDISK.SYS will allow either System RAM or extended memory (above 1 Mbyte) to be used as the Virtual Disc. In addition, more than one Virtual disc at a time can be installed in your system.

Virtual Discs have one big advantage and one big disadvantage over flexible or hard discs. Their advantage is speed. Disc operations on a Virtual disc are significantly faster than on a hard disc. The improvement in performance is even more dramatic when compared with flexible disc drives. The disadvantage in using Virtual Discs is that their contents are lost whenever your HP Vectra is reset, or the power turned off.

Since the contents of a Virtual Disc are lost when the system is turned off or reset, files must be transferred to the disc when the system is started, and back to a flexible or hard disc before the system is reset. This task can be accomplished with either the MS-DOS COPY or BACKUP/RESTORE commands. DISKCOPY will not work with a Virtual Disc.

To install a Virtual Disc Drive, insert the command line

```
DEVICE=[<d>:] [<path>]VDISK.SYS[<bbb>] [<sss>]
[<ddd>] [/E [:<m>]]
```

in the file CONFIG.SYS. The optional parameters are defined below.

<bbb> This is the Virtual Disc size in Kbytes. It is specified as a decimal value ranging between 1 and the amount of memory you have in your computer. The default value is 64 Kbytes. VDISK.SYS may adjust the amount of memory actually used for the Virtual Disc as follows:

- If your computer has less than 64 Kbytes of available memory at the time VDISK.SYS is being installed, VDISK.SYS issues an error message and does not install VDISK.
- If the size you specify is either less than 1 Kbyte or greater than the amount of memory in your computer, VDISK uses the default values of 64 Kbytes.
- If VDISK.SYS adjusts the size entered as <bbb>, a message will be displayed.

<sss> This is the sector size in bytes. Sector sizes of 128, 256, or 512 bytes are permitted. If this parameter is omitted, or if an incorrect value is entered, VDISK.SYS uses the default value of 128 bytes. If you are going to use your VDISK to store many small files, the smaller sector sizes will give you better space efficiency. The larger sector sizes will give you better performance, however. If VDISK.SYS adjusts the sector size entered, a message will be displayed.

<ddd> This is the number of directory entries the VDISK can contain in its Root Directory. <ddd> is specified as a decimal value ranging from 2 through 512. The default value is 64. VDISK.SYS may adjust the values you enter as follows:

- The value is adjusted upward to the nearest sector size boundary. For example, if your sector size is 256 and you specify a value of 17, VDISK.SYS generates 24 directory entries. (Seventeen entries at 32 bytes per entry equals 544 bytes, which is not an even multiple of your sector size. Twenty-four entries occupy 768 bytes, which is a multiple of the sector size.



- If you specify a VDISK size that is too small to hold the file allocation table, the Root Directory, and two additional sectors, the directory size is adjusted downward one sector at a time until these conditions are met. If the directory size reaches 1 sector and the conditions still cannot be met, VDISK.SYS issues an error message and the VDISK is not installed.
- If VDISK.SYS adjusts the number of directory entries, a message will be displayed.

`/E` This option instructs VDISK.SYS to create the VDISK in extended memory (memory at or above 1 Mbyte), although the driver code will still be installed in low memory.

This parameter is valid only for HP Vectras equipped with optional extended memory. If you specify the parameter, and the memory is not present, VDISK.SYS issues an error message and the VDISK is not installed.

`< m >` This is the maximum number of data sectors (of size `< sss >`) that VDISK.SYS transfers back and forth from extended memory at one time. Acceptable values for `< m >` range from 1 through 8, and the default value is 8. This parameter is only valid when the `/E` option has been specified.

When VDISK is operating in extended memory, interrupt servicing is suspended during data transfers. If frequent interrupts occur during data transfers, some of the interrupts may be lost. This might be a problem in certain data communication or other applications. Reduce the value of `< m >` until no interrupts are lost. If an `< m >` of 1 and

<sss> of 128 does not improve the situation, then you cannot use VDISK in extended memory in that particular environment. If VDISK is installed in System memory, and the problem of lost interrupts is not solved, investigate other areas that might be involved.

More than one VDISK may be installed by inserting additional DEVICE=VDISK.SYS commands in the CONFIG.SYS file. Each VDISK increases the resident size of MS-DOS by 720 bytes. Once a VDISK has been installed, the following sign-on message will appear each time MS-DOS is booted.

```
VDISK Version 2.0 virtual disk x
```

where x is the drive designator assigned to the VDISK. Below this message the values of the three parameters are displayed as shown.

```
Buffer size:      <bbb>  
Sector size:     <sss>  
Directory entries: <ddd>
```


Standard Input and Output

- Standard Input and Output Devices
- Redirecting Standard Input and Output
- Piping Standard Input and Output
- Filters
- How to Use Redirectors, Piping, and Filters

8

Standard Input and Output

In this chapter we will examine the Standard Input and Standard Output devices in detail. We'll discuss redirecting and piping program input and output, as well as the use of filters.

Standard Input and Output Devices

We introduced the Standard Input and Output devices in the chapter titled *Understanding MS-DOS Concepts*. Throughout the rest of this chapter we'll be discussing the use of the Standard I/O devices. To start, we'll review a few of the pertinent characteristics of these devices.

When the HP Vectra is booted, the keyboard is assigned as the Standard Input device and the display screen as the Standard Output device. The keyboard and display screen together are referred to as the Console (CON) device.

The Standard Input and Output devices may be assigned to physical devices other than CON via the MS-DOS CTTY command. One of the more common assignments is to the AUX device. This allows a remote terminal to act as the Standard Input and Output devices.

Note

The devices must be assigned as a pair. It is not possible to assign the Output device to AUX while the input device remains assigned to CON.

In addition to the physical devices which are part of MS-DOS, the Standard Input and Output devices may also be assigned to an installed device driver. The driver must be a character device capable of both input and output. Installing device drivers was discussed in detail in the chapter titled *System Configuration*.

Redirecting Standard Input and Output

It is often advantageous to direct Standard Output to a file or device other than the one currently assigned, or to receive Standard Input from a file or other device. This process is called redirection. MS-DOS provides capabilities for redirection of both the Standard Input and Output devices.

Standard Input and Output may be redirected to or from a file as well as another device. Either Standard Input or Output may be redirected independently of the other. Redirection is temporary; it is in effect only until the program or MS-DOS command completes and returns to the system prompt.

Redirecting Standard Input and Output has many applications. Output from a program may be sent to the printer to produce a hard-copy record, or to a file for future use. A program's keyboard input may be placed in a file where it functions somewhat like a Batch Command file.

Redirection of either Standard Input or Output is invoked from the MS-DOS command line. The syntax for redirection is shown below.

```
>[<d>:] [<path>]<filename>[.<ext>]
```

This redirects the Standard Output to either a file or device. If the output is to a file MS-DOS will open or create a file with the name specified, and store all characters sent to the Standard Output in the file. If the file doesn't already exist on the disc, it will be created. If it does, the previous contents will be overwritten.

```
> <device name>
```

Redirecting output to a device (second syntax example) sends all characters to the device named in the command line. Output may be redirected to one of the other Standard Devices (Printer or Auxiliary), or to a physical device (LPT1, COM1, etc.).

The following two examples redirect the output of the MS-DOS DIR command, first to the file DISCDIR.LST, then to the Standard Printer device PRN:

```
A>DIR >DISCDIR.LST
```

```
A>DIR >PRN
```



```
>>[<d>:] [<path>]<file name>
```

This form of redirection of Standard Output is similar to the previous one. The difference is that it appends data to an existing file, instead of overwriting its previous contents. If the file specified in the command line does not exist, it will be created. In this instance it behaves exactly as in the previous example. This form is also valid for redirection to a device. However it performs exactly the same as above.

The command line

```
C>DIR >>DISCDIR.LST
```

will append the output from the MS-DOS DIR command to the previous contents (if any) of the file DISCDIR.LST.

```
<[<d>:] [<path>]<filename>[.<ext>]
```

This redirects the Standard Input device. If a file is specified MS-DOS will read characters from the file as they are requested by the program or MS-DOS command, until the program terminates. It is important when using this form of redirection that all input required by the program be contained in the file. If the program attempts to read more input than is contained in the file, the program will stop, and you'll need to exit by pressing **CTRL]-Break**.

```
< <device name>
```

If a device is specified (second syntax example), MS-DOS will read characters from the specified device.

The following two examples redirect the Standard Input from the file PROGRAM.IN and the AUX device respectively to the application program MYPROG

```
A>MYPROG <PROGRAM.IN
```

```
A>MYPROG <AUX
```

Piping Standard Input and Output

Piping is an extension of redirection. Piping allows two or more programs to be "chained" together; the output of each program serving as the input to the next, as shown in Figure 8-1.

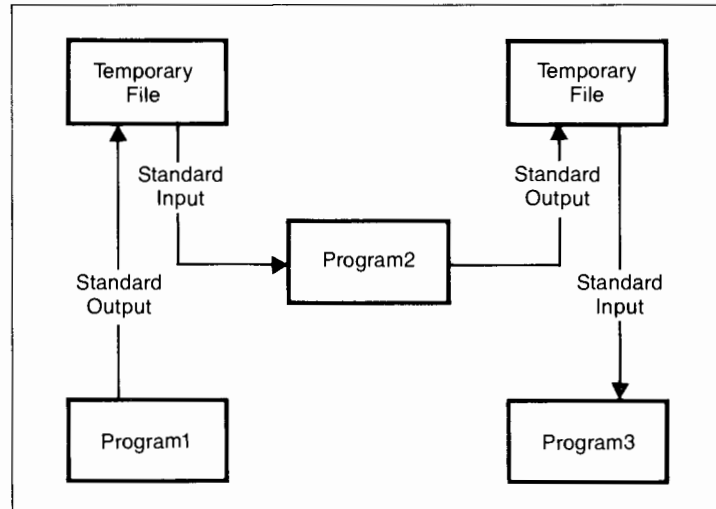


Figure 8-1. Temporary Files Used During Piping

Piping is invoked when MS-DOS encounters two or more programs and/or command names separated by the "|" character in the command line. The following command line will "pipe" the output of PROGRAM1 to the input of PROGRAM2.

```
C>PROGRAM1 | PROGRAM2
```


When piping is invoked, MS-DOS runs the programs in the order they are encountered in the command line; it does not run the programs concurrently as one might expect. A temporary file is created to store the output of the first program, in essence redirecting the Standard Output to a file. The next program redirects its Input from the temporary file. This process is repeated for as many programs as are piped together. When MS-DOS has completed executing the command line, it erases the temporary files. The use of temporary files is illustrated in Figure 8-1.

Filters

The final topic relating to Standard Input and Output is filters. A filter is any program that reads data from the Standard Input device, modifies or examines it, then outputs it to the Standard Output device.

Three filters are provided with MS-DOS: FIND, MORE, and SORT. FIND searches the Standard Input for a specified string of characters; MORE displays input from the Standard Input device one screen at a time; and SORT sorts Standard Input into ascending or descending order, then outputs it to the Standard Output device. These filters are explained in greater detail in the chapter titled *MS-DOS Command Descriptions*.

Let's look at a couple of examples of how filters can be used. The following command line pipes the output of the DIR command through the filter MORE. This is the functional equivalent to using the /P option with DIR; the directory listing pauses after each screenful until you press any key.

```
A>DIR | MORE
```

As we mentioned earlier, more than one command or program may be "chained" together via piping. The following

command line will pipe first to the filter SORT to sort the directory listing into alphabetical order, then pipe to MORE to display the sorted listing one screen at a time.

```
C>DIR | SORT | MORE
```

You may notice one or two file names which you don't remember being on the disc. These are the temporary files used by MS-DOS for piping. Although they will show up in the directory listing produced with the command line above, if you execute DIR afterward (without any piping) you'll see that MS-DOS has erased them.

Tips on Using Redirection, Piping, and Filters

There are several points to remember when using Standard Input and Output, redirection, piping, and filters.

- These capabilities may not work with all programs. Standard Input and Output are MS-DOS functions. Therefore, any program which bypasses MS-DOS for character input and output (i.e. using BIOS interrupts, or going to the hardware itself) will not perform these functions.
- In most cases, piping should only be performed with filters. Filters are specially written programs which use only Standard Input and Output, interpret an end-of-file character from the Standard Input device as a "terminate program" command, and are generally programmed to perform correctly as filters. Using programs which don't meet these criteria with piping will lead to unpredictable or unsatisfactory results.
- MS-DOS allows both piping and redirection in the same command line. Earlier we used the example of the SORT filter and DIR command. The following command line

sorts the directory listing and outputs it to the file DIRLIST. Both piping and redirection are used.

```
A>DIR | SORT >DIRLIST
```

It is also permissible to use redirection to devices. Instead of storing the sorted directory listing in the file, we could print it on the system printer using the command line

```
A>DIR | SORT >PRN
```

- MS-DOS sends all error messages to a logical device referred to as the Standard Error Device. Output to this device (i.e. error messages) is not redirected, even though the Standard Input and Output is. Therefore, if a disc read error was encountered during execution of the command line in the last example, the error message would be displayed on the screen, not the system printer.

9

DEBUG

Introduction	9-1
DEBUG Commands	9-1
DEBUG Command Parameters	9-4
Memory Addressing on the HP Vectra	9-7
How to Start DEBUG	9-10
DEBUG Command Descriptions	9-12
The Assemble Command	9-13
The Compare Command	9-16
The Dump Command	9-17
The Enter Command	9-19
The Fill Command	9-21
The Go Command	9-22
The Hex Command	9-25
The Input Command	9-26
The Load Command	9-27
The Move Command	9-29
The Name Command	9-30
The Output Command	9-33
The Procedure Command	9-34
The Quit Command	9-35
The Register Command	9-36
The Search Command	9-39
The Trace Command	9-40
The Unassemble Command	9-42
The Write Command	9-44
Debug Error Messages	9-46



9

DEBUG

Introduction

The DEBUG program provides an environment to test binary and executable object files. Whereas EDLIN is available to alter source files, DEBUG can be used for executable files.

DEBUG is an advanced program; it deals with topics such as 8086 family architecture, assembly language programming, and other topics which are more difficult than other MS-DOS concepts. We assume the reader has some familiarity with these related topics.

DEBUG eliminates the need to reassemble a program to see if a problem has been fixed by a minor change to the code; it allows you to alter the contents of a file or of a CPU register, and then to immediately execute your program to test your changes.

DEBUG Commands

Each DEBUG command consists of a single letter followed by one or more parameters. If you enter a DEBUG command incorrectly, DEBUG will prompt you by reprinting the command line and pointing to the error with the up-arrow character followed by the word "error."

For example:

```
dcx:100 cs:110
      ^Error
```

All DEBUG commands can be aborted at any time by pressing **CTRL-Break**. **CTRL-Num lock** suspends the display of data so that you can read it before it scrolls off the display; and you can restart the display by pressing any alphabetic or numeric key.

Any combination of upper- and lower-case letters may be used in DEBUG commands and their parameters. The DEBUG commands are summarized in the table that follows, and are described in detail later in this chapter.

Table 9-1. DEBUG Command Summary

DEBUG Command	Function
A [<address>]	Assemble
C [<range>]	Compare
D [<range>]	Dump
E <address> [<list>]	Enter
F <range> <list>	Fill
G [= <addr>][<addr>[<addr>...]]	Go
H <value> <value>	Hex
I <value>	Input

Table 9-1. DEBUG Command Summary (Cont'd)

DEBUG Command	Function
L [<code><address></code> [<code><drive></code> <code><rec></code> <code><rec></code>]]	Load
M <code><range></code> <code><address></code>	Move
N <code><filename></code> [<code><arglist></code>]	Name
O <code><value></code> <code><byte></code>	Output
P [= <code><address></code>] [<code><value></code>]	Procedure
Q	Quit
R [<code><register-name></code>]	Register
S <code><range></code> <code><list></code>	Search
T [= <code><address></code>] [<code><value></code>]	Trace
U [<code><range></code>]	Unassemble
W [<code><address></code> [<code><drive></code> <code><rec></code> <code><rec></code>]]	Write

DEBUG Command Parameters

All DEBUG commands, except the Quit command, accept parameters. Parameters may be separated by either the space or comma delimiter, but a delimiter is only required between two consecutive hexadecimal values. Thus, the following commands are equivalent:

```
dcS:100 110
d cS:100 110
d,cS:100,110
```

Parameter definitions are the same for each DEBUG command that uses them. The DEBUG parameters and their definitions are shown below:

Parameter	Definition
<drive>	A one-digit hexadecimal value that indicates which drive a file will be loaded from or written to. These values designate the drives as follows: 0 = A:, 1 = B:, 2 = C:, ...
<byte>	A two-digit hexadecimal value to be placed in or read from an address or register.
<rec>	A 1- to 3-digit hexadecimal value that indicates the logical record number on the disc and the number of disc sectors to be written or loaded. Logical records correspond to sectors.
<value>	Up to four hexadecimal digits that specify a port number; or the number of times a command should be repeated.

<address> A two-part designation consisting of either an alphabetic segment register designation or a four-digit segment address plus an offset value. The segment designation or segment address may be omitted, in which case the default segment is used. DS is the default segment for all commands except Go, Load, Trace, Unassemble, and Write, for all of which the default segment is CS. All numeric values are hexadecimal. For example:

```
CS:0100  
04BA:0100
```

The colon is required between a segment designation (whether numeric or alphabetic) and an offset. See the next section entitled *Memory Addressing on the HP Vectra* for more information on the address parameter.

<range> Two addresses, as in <address> <address>; or one address followed by "L" and a value, as in <address> L <value>. <value> specifies the number of lines the command should operate on and L80 is assumed. This second syntax for <range> cannot be used if another hex value follows the range because the hex value would be interpreted as the second address of the range. For example:

```
CS:100 110  
CS:100 L 10  
CS:100
```

are all legal specifications for <range>. The following is illegal:

```
CS:100 CS:110
      ^Error
```

The maximum value for <range> is 10000 hex. To specify a value of 10000 hex within four digits, type 0000 (or 0).

<list>

A series of byte or string values. <list> must be the last parameter on a command line. For example:

```
fcS:100 42 45 52 54 41
```

<string>

Any number of characters delimited by either single or double quotation marks. Quotation marks within a delimited string must be doubled. For example:

```
'This is a "string" is legal'
'This is a ''string'' is legal"
"This is a ""string"" is legal"

'This 'string' is not legal'
"This "string" is not legal"
```

Note that double quote marks are not necessary in the following strings:

```
"This ''string'' is not needed"
'This ""string"" is not needed"
```

The ASCII values of the characters in the string are used as a <list> of byte values.

Memory Addressing on the HP Vectra

Before you run DEBUG, it will help you to understand how addresses are stored in the system's memory.

Every byte of RAM has a unique address which may be stored in up to four bytes (which equal 2 words). This format allows 65,536 unique addresses. To increase the number of possible addresses in RAM, the 8086 family of processors employ an algorithm in which the first word is multiplied by 10 (Hex) and added to the second word; this allows 1,048,576 unique addresses.

To accommodate the increased size of RAM and improve its management, four special registers were created in the 80286 processor. These registers hold the first word so that programmers can, in most cases, deal only with the second word called the "offset."

Each of these special registers holds the first address of a "segment" of memory that contains up to 65,536 bytes of RAM. For this reason, these registers are called "segment registers". Each register has a special purpose as indicated by its name: CS (code segment), DS (data segment), SS (stack segment), and ES (extra segment).

If you load a program (code) into memory, the code segment register will contain the first word of the address. For example, if the address in the code segment register is 20A2, you may view the first portion of your program by entering either of the following commands:

```
dc s:100 (offset)
d 20A2:100 (address + offset)
```

In the above examples,

d represents the DEBUG Dump command (explained in more detail later in this chapter)

cs refers to the code segment register

20A2H is the address of the code segment register, and

100H is the offset.

In some instances, you do not need to use either the segment name or its address and offset in conjunction with a specific DEBUG command. The Load command, for example, automatically loads at a default address; and when you enter the Dump command without an address specification, you will see the first portion of your program displayed on your screen just as if you had entered `dc s:100`. See individual command descriptions for more details.

Using the actual address rather than naming the register (i.e., 20A2 rather than cs) can cause problems. For example, if you load to an address instead of using the default, you can overwrite the operating system, or anything else in memory, while using DEBUG. For this reason, it is a good idea to specify a register or use the default.

Caution



Most DEBUG commands, when used incorrectly, can overwrite critical areas of memory and “crash” your HP Vectra System. If you do make such an error, you can usually recover by performing one of the following steps:

1. Perform a `CTRL-ALT-DEL` reset.

If this does not work,

2. Turn your system off, then on, and reload the operating system.

We strongly recommend that you read each command description carefully before using it.

How to Start DEBUG

The syntax to run DEBUG is:

```
DEBUG
```

or

```
DEBUG [<file name> [<arglist>]]
```

When you enter the DEBUG command without parameters, DEBUG responds with:

```
-
```

Because no file name has been specified, current memory, disc sectors, or disc files can be worked on using any of the DEBUG commands.

Caution



When DEBUG is started, it sets up a program header at offset 0 in the program's work area. You can overwrite this header without consequences if no <file name> is given to DEBUG. If you are debugging a .COM or .EXE file, however, do not tamper with the program header below address 5CH (the H stands for Hexadecimal, and will be used throughout this chapter). If you do, DEBUG may not be able to terminate correctly when you quit. Do not attempt to restart a program after the message

```
Program terminated normally
```

is displayed. You must reload the program using the Name and Load commands for it to run properly.

You can also start DEBUG using a command line, as in

```
DEBUG [[<d>:][<path>]<file name>[<arglist>]]
```

When you use this method, the file you specify will be loaded into memory. MS-DOS determines the lowest segment after the end of its resident portion. MS-DOS will load a Program Segment Prefix (PSP) at this address. Refer to the MS-DOS Programmer's Reference Manual for details about the PSP. The PSP is 100H bytes long. The program file is then loaded in after it.

If the file has an extension of either .EXE or .HEX, the DS and ES segment registers point to the PSP, and the CS, SS, SP, and IP registers are set to the value specified in the .EXE or .HEX file. .HEX files are assumed to be in Intel .HEX format and are converted into executable format as they are loaded.

For all other files, all segment registers point to the PSP, the IP register is set to 100H, and the SP register is set to FFEH.

DEBUG also places the number of bytes in the program as a doubleword value in BX:CX. The Least Significant word is placed in CX, the Most Significant word in BX.

The initial values for the flags are:

```
NV UP EI PL NZ NA PO NC
```



An <arglist> may be specified only if <file name> is present. The <arglist> is a list of filename parameters and switches that are to be passed to <file name>. Thus, when <file name> is loaded into memory, it is loaded as if it had been started with the command:

```
A><file name> <arglist>
```

Here, <file name> is the file to be debugged, and the <arglist> is the command line to be passed to the program when it is executed.

DEBUG Command Descriptions

All debug commands are described in detail in the pages that follow. Each DEBUG command consists of a single letter followed by one or more parameters. DEBUG commands can be edited using the Editing Keys described in the chapter titled *Using the MS-DOS Keyboard*.

We strongly recommend that you read information on DEBUG commands before you attempt to use them. Many commands contain important notes and cautions.

The Assemble Command

Purpose

The Assemble command assembles 8086, 8087, and 8088 mnemonics directly into memory.

Syntax

```
A [<address>]
```

Operation

All numeric values are hexadecimal and must be entered as 1–4 characters. Prefix mnemonics (such as JMP for Jump) must be specified in front of the opcode to which they refer; or they may be entered on a separate line.

The segment override mnemonics are CS:, DS:, ES:, and SS:. The mnemonic for the far return is RETF. String manipulation mnemonics must explicitly state the string size. For example, use MOVSW to move word strings, and MOVSB to move byte strings.

The assembler will automatically assemble short, near, or far jumps and calls, depending on the byte displacement to the destination address. These may be overridden with the NEAR or FAR prefix. For example:

```
0100:0500 JMP 502      ; a 2-byte short jump
0100:0502 JMP NEAR 505 ; a 3-byte near jump
0100:0505 JMP FAR 50A  ; a 5-byte far jump
```

The NEAR prefix may be abbreviated to NE, but the FAR prefix cannot be abbreviated.

DEBUG cannot tell whether some operands refer to a word memory location or to a byte memory location. In such a case, the data type must be explicitly stated with one of the prefixes "WORD PTR" or "BYTE PTR." Acceptable abbreviations are "WO" and "BY" respectively. For example:

```
NEG    BYTE PTR <128>
DEC    W0 [SI]
```

Nor can DEBUG tell whether an operand refers to a memory location or to an immediate operand. DEBUG uses the common convention that an operand enclosed in square brackets refers to a memory location. For example:

```
MOV    AX,21        ;Load AX with 21H
MOV    AX,[21]      ;Load AX with contents
                        ;of location DS:21H
```

Two popular pseudo-instructions are available with Assemble. The DB opcode assembles byte values directly into memory; the DW opcode assembles word values directly into memory. For example:

```
DB     1,2,3,4,"THIS IS AN EXAMPLE"
DB     'THIS IS A QUOTE:'
DB     "THIS IS A QUOTE:"

DW     1000,2000,3000,"BACH"
```

Assemble supports all forms of register indirect commands. For example:

```
ADD    BX,34[BP+2] . [SI-1]
POP    [BP+DI]
PUSH   [SI]
```

All opcode synonyms are also supported. For example:

```
LOOPZ 100
LOOPE 100
```

```
JA    200  
JNBE 200
```

Press the key without typing an op code to terminate the Assemble command; you can terminate the command at any byte position.

The Compare Command

Purpose

The Compare command compares the portion of memory specified by <range> to a portion of the same size beginning at a specified address.

Syntax

```
C> <range><address>
```

Operation

If <range> and <address> specify identical areas of memory, no comparison is made and you are returned to the DEBUG command prompt.

If <range> and <address> are different, the non-matching bytes are displayed in the following format:

```
<address1> <byte1> <byte2> <address2>
```

Example

To compare the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH, enter:

```
C100,1FF 300
```

or

```
C100L100 300
```

The two commands have the same effect.

The Dump Command

Purpose

The Dump command displays the contents of the specified region of memory.

Syntax

D [<range>]



Operation

If you enter the D command without parameters, 128 bytes are displayed at the first address after the address displayed by the previous Dump command. If a range of addresses is specified, the contents of the range are displayed.

The dump is displayed in two portions: a hexadecimal dump with each byte shown as a hexadecimal value, and an ASCII dump with bytes shown as their corresponding ASCII characters. Nonprinting characters are denoted with a period (.) in the ASCII portion of the display. Each display line shows 16 bytes with a hyphen between the eighth and ninth bytes. Each displayed line begins on a 16-byte boundary.

If you enter the command:

```
dc:0100 L 20
```

or

```
cs:0100 011F
```

DEBUG displays the dump in the following format:

```
207E:0100 48 50 20 56 b5 b3 74 72-b1 20 54 bF 75 b3 b8 20 HP Vectra
207E:0110 53 b3 72 b5 b5 bE 00 00-00 00 00 00 00 00 00 Screen...
```

If you type the following command:

```
D
```

the display is formatted as described above. Each line begins with an address, incremented by 16 from the address on the previous line. Each subsequent D command entered without parameters displays the bytes immediately following those last displayed.

If you type the command:

```
DCS:100 L 20
```

the display will be formatted as described above, but 20H bytes will be displayed. If you then type the command:

```
DCS:100 115
```

the display will still be formatted in the same way but only the bytes in the range of lines from 100H to 115H in the CS segment will be displayed.

The Enter Command

Purpose

The Enter command enters byte values into memory at the specified address.

Syntax

```
E <address> [<list>]
```

Operation

If <address> is specified without the optional list, DEBUG displays the address and its contents, then waits for your input. At this point, DEBUG expects you will perform one of the following actions:

1. Replace a current byte value with a new value by entering the new value after the current value. If the value you enter is not a legal hexadecimal value, or if more than two digits are typed, the illegal extra character(s) are not echoed.
2. Use the `space bar` to advance to the next byte; to change its value, simply type the new value after the current value. If you space beyond an 8-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.
3. Type a hyphen (-) to return to the preceding byte. If you decide to change a byte prior to the current position, the hyphen returns the current position to the previous byte, and a new line is started with the address and its byte value displayed.
4. Press `Enter` to terminate the Enter command. You can press `Enter` to terminate the command at any byte position.

If the optional list of values is entered with the Enter command, byte values are replaced automatically with new values. If an error occurs, no byte values are replaced.

Example

If you were to enter the following DEBUG command:

```
ECS:100
```

DEBUG might respond by displaying

```
04Ba:0100 EB. _
```

(The underscore represents the cursor.) To change this value to 41, type 41 as shown below:

```
04BA:0100 EB.41_
```

To step through the subsequent bytes, press the to see:

```
04BA:0100 EB.41 10. 00. BC. _
```

To change BCH to 42H, enter:

```
04BA:0100 EB.41 10. 00. BC.42_
```

Now, to change 10H to 6FH, type the hyphen as many times as needed to return the cursor to byte 0101H (value 10H), then replace 10H with 6FH:

```
04BA:0100 EB.41 10. 00. BC.42
04BA:0102 00.-
04BA:0101 10.6F_
```

Press to terminate the Enter command and return to the DEBUG command prompt.

The Fill Command

Purpose

The Fill command fills the addresses in the specified range with the values specified in the list.

Syntax

```
F <range><list>
```

Operation

If the specified range contains more bytes than the number of values in the specified list, the list will be used repeatedly until all bytes in the range are filled.

If the list contains more values than the number of bytes in the range, the extra values in the list will be ignored. If any of the memory in the range is not valid (bad or non-existent), you will receive an error message. If a segment value is not specified in the range, DS is assumed.

Example

Assume that you have entered the following DEBUG command:

```
F04BA:100 L 100 42 45 52 54 41
```

DEBUG fills memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values are repeated until all 100H bytes are filled.

The Go Command

Purpose

The Go command executes the program currently in memory.

Syntax

```
G [= <address>] [<address> [<address>...]]
```

Operation

If the Go command is entered without parameters, the program in memory will be executed as if it had been executed from the MSDOS prompt.

If the = <address> parameter is set, execution begins at the specified address. The equal sign (=) is required so that DEBUG can distinguish the start address from breakpoint address(es).

When additional addresses are set, program execution stops at the first of those addresses encountered, regardless of its position in the address list. When program execution reaches a breakpoint, the registers, flags, and decoded instructions are displayed for the last instruction executed. The result is the same as if you had typed the Register command for the breakpoint address.

Up to ten breakpoints may be set. Breakpoints may be set only at addresses containing the first byte of an 80826 opcode. If more than ten breakpoints are set, DEBUG will return the BP error message.

The user stack pointer must be valid and have six bytes available for the Go command. The Go command uses an IRET instruction to cause a jump to the program under test. The user stack pointer is set, and the user flags, Code Segment register, and Instruction Pointer are pushed on the user stack.

Thus, if the user stack is not valid or is too small, the operating system may crash. An interrupt code (OCCH) is placed at the specified breakpoint address(es).

When an instruction with the breakpoint code is encountered, all breakpoint addresses are restored to their original instructions. If execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

Example 1

Assume that you have entered the following command:

```
GCS:7550
```

The program currently in memory executes up to the address 7550 in the CS segment. DEBUG then displays registers and flags, and then terminates the Go command.

If you re-enter the Go command after a breakpoint has been encountered, the program in memory restarts at the location of the breakpoint.

When you execute all or part of a program using Go, the registers and memory may change; to run the program again, you should reload the program using the Load command to reinitialize the registers and memory.

Example 2

Assume the IP is set to 100H, the command

```
G 200
```

starts program execution at IP and stops at 200H.



If you had entered

G=100

execution would have begun at 100H and continued to the end of the program.

If you had entered

G=100 200H

execution would have begun at 100H and continued to 200H.

The Hex Command

Purpose

The Hex command performs hexadecimal arithmetic on the two parameters specified.

Syntax

```
H <value><value>
```

Operation

This command is useful in calculating offsets from current or known addresses. DEBUG first adds the two specified values, then subtracts the second value from the first value. The results of these operations are displayed on one line; first the sum, then the difference.

Example

You have entered the following DEBUG command:

```
H19F 10A
```

DEBUG first sums the two values, then subtracts 10AH from 19FH. The following result is displayed:

```
02A9 0095
```

The Input Command

Purpose

The Input command inputs and displays one byte from the port specified by the value parameter.

Syntax

```
I <value>
```

Operation

A 16-bit port address may be specified as the value.

Example

Assume that you have entered the following command:

```
I2F8
```

and that the byte at the port is 42H. DEBUG inputs the byte and displays the following value:

```
42
```

Because HP Vectra programmers can use MS-DOS function calls inside their programs, you should not find it necessary to use the Input command.

The Load Command

Purpose

The Load command loads a disc file or absolute disc sectors into memory.

Syntax

```
L [<address> [<drive><rec><rec>]]
```

Operation

To debug a disc file rather than absolute disc sectors, the file to be loaded must have been named either when DEBUG was started, or using the DEBUG Name command. Both the command to start DEBUG and the Name command create a non-extended File Control Block at CS:5C. BX:CX is the number of bytes read.

If you enter the Load command without any parameters, DEBUG loads the named file into memory beginning at address CS:100 and sets BX:CX to the number of bytes loaded.

If the Load command is entered with just an address parameter, loading begins at the memory address specified.

If L is entered with all the parameters, absolute disc sectors are loaded rather than a file. The specified records are taken from the specified drive and DEBUG begins loading with the first record specified, and continues until the number of sectors specified in the second record have been loaded.

Example

Assume that the following commands have been entered:

```
B>DEBUG
-NFILE.COM
```

NFILE.COM is the DEBUG command to Name the file to be loaded into memory. Now, to load FILE.COM, enter:

```
L
```

DEBUG loads FILE.COM and displays the DEBUG prompt. Assuming that you want to load only portions of a file or certain records from disc, you can enter:

```
L04BA:100 2 OF 6D
```

DEBUG will then load 109 (6D hex) records from drive 2 (C:) beginning with logical record number 15 (0FH) into memory starting at address 04BA:0100. When the records have been loaded, DEBUG again returns the “ - ” prompt.

If the file to be loaded has an .EXE extension, it is relocated to the load address specified in the header of the .EXE file; the address parameter is always ignored for .EXE files. The header itself is stripped from the .EXE file before it is loaded into memory; thus the size of an .EXE file on disc will differ from its size in memory.

If the file named with the Name command or specified when DEBUG is started is a .HEX file, then entering the L command with no parameters causes DEBUG to load the file beginning at the address specified in the .HEX file. If the L command includes the address parameter, DEBUG adds the specified address to the address found in the .HEX file to determine the start address for loading the file. Registers BX:CX contain the a doubleword count of the number of bytes in the program. The Least Significant word is stored in CX, the Most Significant word in BX.

The Move Command

Purpose

The Move command moves the block of memory in the specified range to the location beginning at the specified address.

Syntax

```
M <range><address>
```

Operation

If you instruct DEBUG to make an “overlapping” move (that is, one in which part of the block overlaps some of the current address) the move will be performed without loss of data because addresses that could be overwritten are moved first.

The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block’s lowest address, then to work toward the highest.

The sequence for moves from lower addresses to higher addresses is to move the data beginning at the block’s highest address and to work toward the lowest.

Example

Assume that you enter:

```
MCS:100 110 CS:500
```

DEBUG will first move address CS:110 to address CS:510; then CS:10F to CS:50F, and so on until CS:100 is moved to CS:500. To see the results of the Move, use the Dump command and specify the address specified for the Move command. DEBUG will assume the segment of all addresses to be DS unless a segment is specified.

The Name Command

Purpose

The Name command sets a filename for a later Load or Write command, and/or sets filename parameters.

Syntax

```
N <filename> [<filename> ... ]
```

Operation

If you start DEBUG without naming a file to be debugged, you must use the Name command before a file can be loaded. The Name command also accepts parameters that are used by the file being debugged.

Assume you have entered the following commands:

```
B>DEBUG  
-NFILE1.EXE  
-L  
-G
```

The Name command performs or enables the following steps:

1. Name sets FILE1.EXE as the filename to be used in any later Load or Write commands during the current DEBUG session.
2. Name also sets FILE1.EXE as the first parameter used by any program that is later debugged during the current DEBUG session.
3. The Load command loads the named FILE1.EXE into memory.

4. The Go command causes FILE1.EXE to be executed with FILE1.EXE as the single filename parameter; that is, FILE1.EXE is executed as if

```
FILE1 FILE1.EXE
```

had been typed at the MS-DOS command prompt.

A more useful group of commands might look like this:

```
B>DEBUG  
-NFILE1.EXE  
-L  
-NFILE2.DAT FILE3.DAT  
-G
```



In this series of commands, Name sets FILE1.EXE as the filename to be used by the subsequent Load command. The Load command loads FILE1.EXE into memory, and the Name command is used again, this time to specify the parameters to be used by FILE1.EXE. Finally, when the Go command is executed, FILE1.EXE is executed as if the following command had been entered:

```
B>FILE1 FILE2.DAT FILE3.DAT
```

Note



If a Write command were executed at this point, then FILE1.EXE, the file being debugged, would be saved with the file name FILE2.DAT. To avoid this, you should always execute a Name command before either Load or Write.

Four regions of memory can be affected by the Name command:

```
CS:5C FCB for file 1  
CS:6C FCB for file 2  
CS:80 Count of characters  
CS:81 Unparsed command line
```

A File Control Block (FCB) is set up at CS:5C for the first filename parameter given to the Name command. If a second parameter is entered, then an FCB is set up for it beginning at CS:6C.

The number of characters in the Name command, exclusive of the "N" itself, is given at location CS:80. The actual stream of characters in the Name command, again exclusive of the letter "N," begins at CS:81. Note that this stream of characters may contain switches and delimiters that would be legal in any command typed at the MS-DOS command level.

Example

A typical use of the name command follows:

```
B>DEBUG PROG.COM
-NPARAM1 PARAM2/C
-G
-
```

In this instance, the Go command executes the file in memory as if the following command line has been typed from MS-DOS:

```
B>PROG PARAM1 PARAM2/C
```

Testing and debugging therefore reflect a normal runtime environment for PROG.COM.

The Output Command

Purpose

The Output command sends the specified byte to the specified output port.

Syntax

```
o <value> <byte>
```

Operation

A 16-bit port address is allowed in the output command. If you enter:

```
o2f8 4f
```

DEBUG will output the byte value 4F to output port 2F8.

Because HP Vectra programmers can use MS-DOS function calls inside their programs, you should not find it necessary to use the Output command.

The Procedure Command

Purpose

The Procedure command executes one instruction and displays the contents of all registers and flags and the decoded instruction.

Syntax

```
P [=<address>] [<value>]
```

Operation

The Procedure command operates exactly as the Trace command with the exception that when a subroutine is encountered during a trace operation, it is executed and the trace is resumed at the first line following the subroutine.

This command is most useful to skip over either a tested or well known procedure in order to see the returned values of the next instruction in the current procedure. One such example might be to execute MS-DOS function calls by tracing to INT 21H, (the MS-DOS interrupt for an MS-DOS Function call) then to type P and execute the function. DEBUG will return at the next instruction in your sequence.

See the section on the Trace command later in this chapter for details on the operation of this command.

The Quit Command

Purpose

The Quit command terminates the DEBUG utility and returns the user to the MS-DOS command prompt.

Syntax

q

Operation

The Quit command uses no parameters and terminates DEBUG without saving the file currently in memory. You are returned to the MS-DOS command level.

Example

To end the current debugging session, enter:

q

DEBUG will terminate, and control will return to MS-DOS.

The Register Command

Purpose

The Register Command displays the contents of one or more CPU registers.

Syntax

```
R [<register-name>]
```

Operation

When DEBUG is started, the segment registers are set to the bottom of free memory, the Instruction Pointer is set to 0100H, all flags are cleared, and the remaining registers are set to zero.

If no register-name parameter is used, the Register command dumps the register save area and displays the contents of all registers and flags.

If a register-name is entered, the 16-bit value of that register is displayed in hexadecimal, followed by a colon (:). You can then either enter a <value> to change the register, or you can press if no change is needed.

Valid register-names are:

AX	BP	SS	
BX	SI	CS	
CX	DI	IP	(IP and PC both refer
DX	DS	PC	to the Instruction
SP	ES	F	Pointer.)

Any other entry for register-name will result in a BR error message (see the section entitled *DEBUG Error Messages* for an explanation).

If F is used as the register-name, DEBUG displays each flag with a two-character alphabetic code. To alter any flag, type the opposite two-letter code.; the flags are either set or cleared. The flags are listed in the following table with their codes for SET and CLEAR:

FLAG NAME	SET	CLEAR
Overflow	OV	NV
Direction	DN Decrement	UP Increment
Interrupt	EI Enabled	DI Disabled
Sign	NG Negative	PL Plus
Zero	ZR	NZ
Auxiliary	AC	NA Carry
Parity	PE Even	PO Odd
Carry	CY	NC

When you type the command "RF", the flags are displayed in a row at the beginning of a line in the order shown above. At the end of the list of flags, DEBUG displays a hyphen (-); you may enter new flag values as alphabetic pairs in any order. You do not have to leave spaces between the flag entries.

If more than one value is entered for a flag, DEBUG returns a DF error message. If you enter a flag code other than those shown above, DEBUG returns a BF error message. In both cases, the flags up to the error in the list are changed; flags at the point of the error and following are not changed.

To exit the R command, press the key. Flags for which new values were not entered remain unchanged.

Example

If you enter:

```
R
```

DEBUG displays all registers, flags, and the decoded instruction for the current location. If the location is CS:11A, then the display will look similar to this:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000 SI=005C DI=0000
DS=04BA ES=04BA SS=04BA CS=04BA IP=011A NV UP DI NG NZ AC PE NC
04BA:011A CD21 INT 21
```

If you enter:

```
RF
```

DEBUG will display the flags:

```
NV UP DI NG NZ AC PE NC> _
```

Now, if you enter a valid flag designation, in any order, with or without spaces, as in:

```
NV UP DI NG NZ AC PE NC>PLEICY 
```

Debug will respond only with the DEBUG prompt. To see the changes, you should enter either the R or RF command to see the following display:

```
RF
NV UP EI PL NZ AC PE CY> _
```

You can press to leave the flags this way, or specify new flag values.

The Search Command

Purpose

The Search command searches the specified range for the specified list of bytes.

Syntax

```
S <range><list>
```

Operation

The list may contain one or more bytes, each separated by a space or comma delimiter. If the list contains more than one byte, only the first address of the byte string is returned. Otherwise, all addresses of the byte in the range are displayed. If no match is found, no address is displayed.

Example

If you enter:

```
SCS:100 110 41
```

DEBUG will display a response similar to the following:

```
04BA:0104  
04BA:010D  
-
```

The Trace Command

Purpose

The Trace command executes one instruction and displays the contents of all registers and flags and the decoded instruction.

Syntax

```
T [=<address>] [<value>]
```

Operation

If the optional = <address> parameter is entered, tracing occurs at the specified address. The optional value causes DEBUG to execute and trace the number of steps specified by <value>.

Because the Trace command uses the hardware trace mode of the 80286 microprocessor, you can also trace instructions stored in Read Only Memory.

Example

When you type

```
T
```

DEBUG displays the state of the registers and flags after the last instruction is executed, and decodes the next instruction to be executed. Assuming that the current position is 04BA:011A, DEBUG might return this information:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000 SI=005C DI=0000
DS=04BA ES=04BA SS=04BA CS=04BA IP=011A NV UP DI NG NZ AC PE NC
04BA:011A CD21 INT 21
```

If you enter:

```
T=011A 10
```

DEBUG executes sixteen (10 hex) instructions beginning at 011A in the current segment, and then displays all registers and flags for each instruction as it is executed.

The display continues to scroll until the last instruction is executed; when scrolling stops, you can see the register and flag values for the last few instructions performed. Remember that **CTRL**-**Num lock** suspends the display at any point to allow you time to study the registers and flags for any instruction. Resume scrolling by pressing any alphabetic or numeric character.

Tracing interrupts is possible, as is tracing INs and OUTs; however, most users will not want to watch each instruction of an MS-DOS function call being executed. For this reason, it is recommended that you use the P (Procedure) command at INT 21H, the MS-DOS interrupt for an MS-DOS function call, and at CALL instructions.



The Unassemble Command

Purpose

The Unassemble command disassembles bytes and displays their corresponding source statements with their addresses and byte values.

Syntax

```
U [<range>]
```

Operation

If you enter the U command without parameters, 20 hexadecimal bytes are disassembled at the first address after the address displayed by the previous Unassemble command.

If you type the U command with the <range> parameter, then DEBUG disassembles all bytes in the range. If the <range> is given as an <address> only, then 20H bytes are disassembled.

The display of disassembled code has the same appearance as a listing for an assembled file.

Example

If you enter

```
U04Ba:100 L10
```

DEBUG disassembles 16 bytes beginning at address 04BA:0100:

```
04BA:0100 206472 AND [SI+72],AH
04BA:0103 69 DB 69
04BA:0104 7665 JBE 016B
04BA:0106 207370 AND [BP+DI+70],DH
04BA:0109 65 DB 65
04BA:010A 63 DB 63
04BA:010B 69 DB 69
04BA:010C 66 DB 66
04BA:010D 69 DB 69
04BA:010E 63 DB 63
04BA:010F 61 DB 61
```

If you enter

```
U04ba:0100 0108
```

you will see the following display:

```
04BA:0100 206472 AND [SI+72],AH
04BA:0103 69 DB 69
04BA:0104 7665 JBE 016B
04BA:0106 207370 AND [BP+DI+70],DH
```

If the bytes in some addresses are altered, the disassembler alters the instruction statements. The U command can be typed for the changed locations, the new instructions viewed, and the disassembled code used to edit the source file.

The Write Command

Purpose

The Write command writes the file being debugged to a disc file.

Syntax

```
W[<address>[<drive><rec><rec>]]
```

Operation

If you enter *W* with no parameters, *BX:CX* must already be set to the number of bytes to be written; the file is written beginning from *CS:100*.

If the *W* command is entered with just an address, then the file is written beginning at that address. Again, *BX:CX* must be set before you use the Write command.

Note that if a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file, so long as the length has not changed. The file must have been named either when *DEBUG* was called, or with the *N* (Name) command, described earlier in this section. Both the *DEBUG* call and the *Name* command format a filename properly in the normal format of a File Control Block at *CS:5C*.

If the W command is entered with parameters, the write begins from the memory address specified; the file is written to the specified drive; DEBUG writes the file beginning with the logical record number specified by the first <rec>; and DEBUG continues to write the file until the number of sectors specified in the second <rec> have been written.

Caution



Writing to absolute sectors is EXTREMELY dangerous because the process bypasses the file handler.



Debug Error Messages

Error Code	Definition
BF	<p>Bad flag</p> <p>You attempted to alter a flag, but the characters typed were not one of the acceptable pairs of flag values. See the Register command for the list of acceptable flag entries.</p>
BP	<p>Too many breakpoints</p> <p>You specified more than ten breakpoints as parameters to the G (Go) command. Retype the Go command with ten or fewer breakpoints.</p>
BR	<p>Bad register</p> <p>You typed the R command with an invalid register name. See the Register command for the list of valid register names.</p>
DF	<p>Double flag</p> <p>You typed two values for one flag. You may specify a flag value only once per RF command.</p>

What MS-LINK is and What It Does 10-1

Definition of Terms 10-1

How MS-LINK Combines and Arranges Segments 10-1

Files that MS-LINK Uses 10-2

Input Files 10-2

Output Files 10-2

Virtual Memory File (VMMMF File) 10-2

Running MS-LINK 10-3

Working With MS-LINK 10-3

Using the Text-Prompt Method 10-3

Using the Command-Line Method 10-3

Using the Response-File Method 10-3

Creating a Response File 10-3

Command Prompts 10-3

Object Modules [.OBJ]: 10-3

Run File [First-Object-filename.EXE]: 10-3

List File [NUL.MAP]: 10-3

Libraries [.LIB]: 10-3

Command Characters 10-3

The Plus-Sign 10-3

The Semicolon 10-3

[CTRL]-C 10-3

Optional Switches 10-3

The /DSALLOCATE Switch 10-3

The /HIGH Switch 10-3

The /LINENUMBERS Switch 10-3

The /MAP Switch 10-3

The /PAUSE Switch 10-3

The /STACK:<number> Switch 10-3

Error Messages 10-3

MS-LINK Examples 10-4

10

MS-LINK

This section describes the MS-LINK utility in an abstract, theoretical way for the benefit of readers who are not already familiar with this or similar programs. Readers who desire only practical examples are encouraged to refer directly to the sections entitled *Running MS-LINK* and *MS-LINK Examples*.

What MS-LINK is and What it Does

The MS-LINK utility is a program that requires a minimum of 50K bytes of memory (40K bytes for code and data, and 10K bytes for run space). It communicates with the user by displaying prompts on the screen, to which the user responds by entering the appropriate commands.

MS-LINK joins modules of 80286 object code that have been created separately by a compiler and/or assembler from the source code written by a programmer. The object-code modules must be in the form of 80286 object-code (.OBJ) files. MS-LINK transforms one or more object modules into one load module (the .EXE file).

As MS-LINK combines the object modules into an .EXE file, it resolves any external references that the object modules make to symbols that are defined in other modules. MS-LINK can then search one or more library files to find the definition of any external references that it has not been able to resolve.

MS-LINK also produces a list file that shows the external references which the program has resolved. This file also contains any error messages that were generated during MS-LINK operations.

The output (RUN) file from MS-LINK is not bound to specific memory addresses. Therefore, this file can be loaded and executed at any convenient address by your computer's operating system.

MS-LINK uses as much available memory as possible. When it has used all of the available memory, MS-LINK creates a disc file and becomes a virtual memory linker.

Figure 10-1 below is a diagram of the relationship between MS-LINK and other elements of the system.

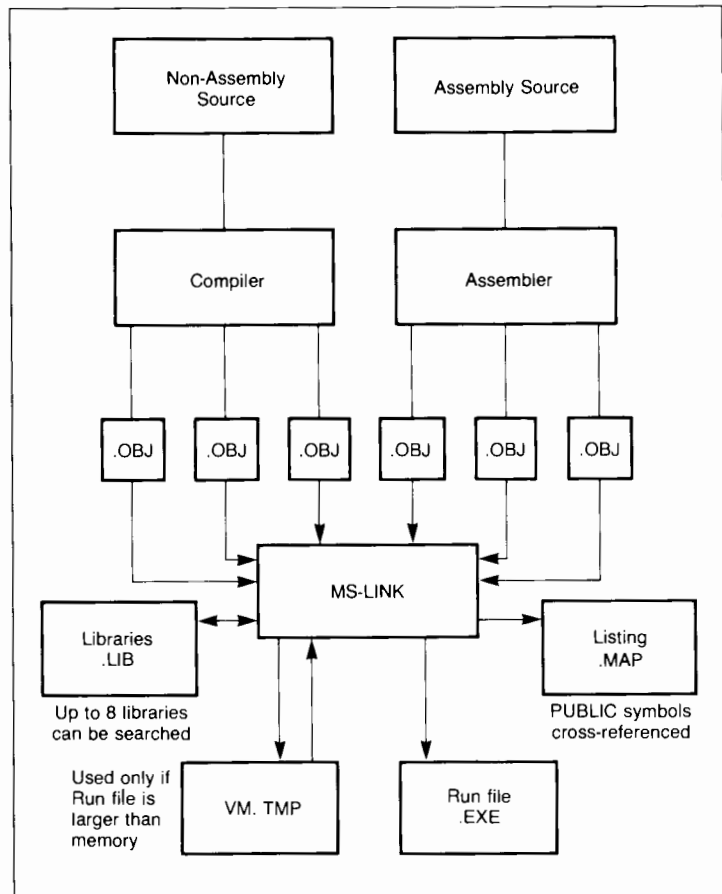


Figure 10-1. MS-LINK in Relation to Other System Elements

Definition of Terms

The terms **segment**, **group**, and **class** appear throughout this discussion. Before continuing, you should understand what these terms mean.

A **segment** is a contiguous area of memory that is up to 64K bytes long. A segment can be located anywhere in 80286 memory adjacent to a paragraph (16-byte) boundary. The contents of a segment are addressed by a segment-register/offset pair.

A **group** is a set of segments that fit within 64K bytes of memory: that is, a collection of segments that are addressed by a single segment register. Either the assembler, the compiler, or the user assigns the group name to the segments. In the latter case, the user assigns the group name in the assembly-language program. For high-level languages, the compiler names the group of segments.

Segments in memory are addressed via their group. Each group is addressed by one segment register. Each segment within a group is addressed by the combination of a segment register and an offset. MS-LINK checks whether the object modules in a group fit within the required 64K bytes of memory. If they don't fit, MS-LINK displays an error message saying so.

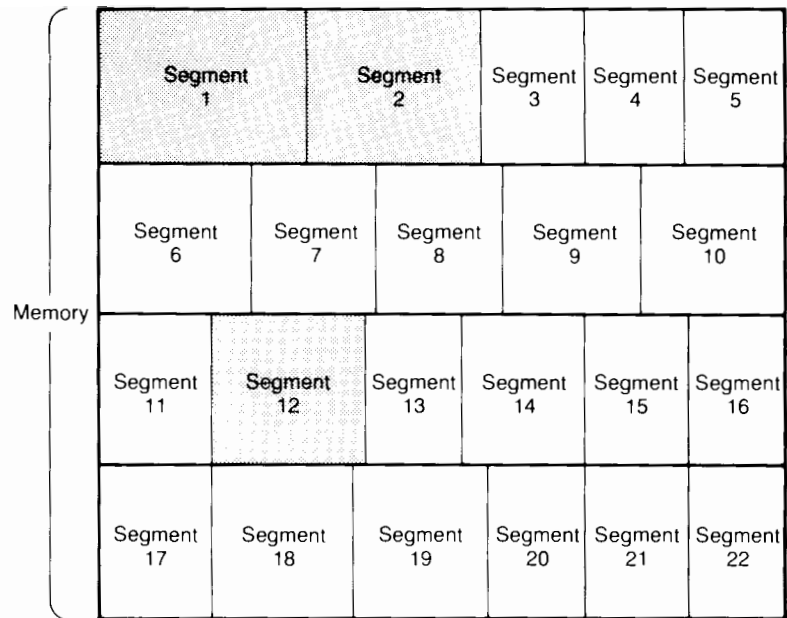
A **class** is a collection of segments. The placement of segments in a class controls the order and relative placement of segments in memory. The segments in a class need not have the same address. The user assigns the class name in an assembly-language program. For high-level languages (such as BASIC, FORTRAN, COBOL, and PASCAL), the compiler assigns the class name. The segments are placed in a class at compile time or at assembly time.

The segments in a class are loaded into memory contiguously. MS-LINK places the segments within a class in the order in which it finds the segments in the object files. A class precedes another class in memory only if a segment in the first class precedes all of the segments in the second class during input to MS-LINK. The classes, which can be loaded across 64Kbyte boundaries, are divided into groups for addressing purposes.

To summarize, to use MS-LINK, you need to remember that:

- a **paragraph** is a memory area that is 16 bytes long,
- a **segment** is a memory area that is up to 64K bytes long,
- a **group** is a set of segments that occupies up to 64K bytes,
- and a **class** is a collection of contiguous segments that can be loaded into memory consecutively across 64Kbyte boundaries.

Figure 10-2 below illustrates the relationship between segments and groups in memory.



Shaded area = A group (64K bytes addressable)

Figure 10-2. Segments and Groups in Memory

How MS-LINK Combines and Arranges Segments

In joining modules of object code, MS-LINK works with the following four types of combinations:

- Private combinations
- Public combinations
- Stack combinations
- Common combinations

These types are declared in the source module for the assembler or the compiler. (The memory-combination type available in the MACRO-86 assembler is treated as a public combination type.) MS-LINK does not automatically place memory-combination types as the highest segments. Instead, MS-LINK generally places segments in the .EXE file that it produces in the same order in which it found the segments in the object modules.

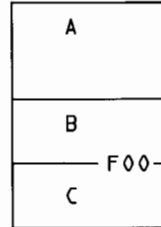
Table 10-1 summarizes how MS-LINK handles segments in private, public, and common combinations. (MS-LINK treats stack combinations very much like public combinations. Because a stack type theoretically has only one segment, combination should not be necessary.)

Table 10-1. How MS-LINK Handles Segments in Private, Public, and Common Types of Combinations

Type	Combination Procedure
Private	Segments are loaded separately, and stay separate. They can be contiguous physically but not logically, even if the segments have the same name. Each private segment has its own base address (that is, a different segment address value).
Public	Segments having the same segment name and the same class name are loaded contiguously, and have the same single base address. The offsets range from the beginning of the first segment through the end of the last segment. (Stack and memory combination types are treated like public types, except that the stack pointer is set to the highest address of the last stack segment.)
Common	Segments having the same segment name and class name are loaded overlapping one another, and have the same single base address. The length of the common area is the length of the longest segment.

As shown in Figure 10-2 (Segments and Groups in Memory), the placement of segments in a group in the assembler lets items be addressed from a single base address for all of the segments in that group.

```
DS: DGROUP XXXX0H     □ -- relative offset
```



Any number of other segments can be placed between the segments in a group. Thus, the offset of FOO can exceed the combined size of all of the segments in the group (up to a total of 64K). An operand of DGROUP:FOO returns the offset of FOO from the beginning of the first segment of DGROUP (above, Segment A).

Segments are grouped according to declared class names. MS-LINK loads all of the segments that belong to the first class name it meets, and then loads all of the segments that belong to the next class name it meets, and so on until it has loaded all of the classes.

For example,

If your assembly program contains the following segments:

```
A SEGMENT 'FOO'  
B SEGMENT 'BAZ'  
C SEGMENT 'BAZ'  
D SEGMENT ZOO'  
  
E SEGMENT 'FOO'
```

They will be loaded in the following order:

```
FOO  
A A, E, B, C,  
E and D are  
BAZ' segment  
names;  
B FOO, BAZ, and  
C ZOO are  
'ZOO' classes.  
D
```

If you are writing an assembly language program, you can control the ordering of classes in memory by writing a dummy module and listing it first in response to the MS-LINK `Object Modules` prompt. In this dummy module, you should declare the segments according to classes in the order in which you want to load the classes.

Caution



Do not use this method with BASIC, COBOL, FORTRAN, or PASCAL programs. Instead, let the compiler and the linker perform their tasks in the normal way.

For example, consider the following module:

```
A SEGMENT 'CODE'  
A ENDS  
  
B SEGMENT 'CONST'  
B ENDS  
  
C SEGMENT 'DATA'  
C ENDS  
  
D SEGMENT STACK 'STACK'  
D ENDS  
  
E SEGMENT 'MEMORY' (9)  
E ENDS
```



Be careful to declare in this module all of the classes to be used in your program. Otherwise, you will lose full control over the ordering of the classes.

You can also use this method if you want to load one or more memory-combined types as the last segments of your program. To do so, add MEMORY between SEGMENT and 'MEMORY' in line (9) above.

It is important to understand that these segments will be loaded last only because you specified this order, not because of any actions that are part of the linking or assembly operations.

Files that MS-LINK Uses

MS-LINK uses the four types of files shown in Table 10-2 below:

Table 10-2. The Types of Files Used by MS-LINK

Type	Description	Filename Extension
Input files	Object-code files	.OBJ
Output files	Executable RUN files	.EXE
	Listing files	.MAP
VM.TMP file	Temporary file	.TMP
Library files	Library, created earlier	.LIB

MS-LINK works with one or more input files (which are the object modules), and produces two output files. MS-LINK can also create a virtual memory file, and can be instructed to search through from zero to eight library files.

You can specify each type of file, according to the following format:

```
[<d>:]<filename>[.<ext>]
```

In this specification,

d:

designates the drive the file is on

filename

is a file name consisting of from 1 to 8 characters

.ext

is a file name extension consisting of a period and from 1 to 3 characters.

Together, these make up the file name. The filename extension is not a mandatory part of the file name.

You can also enter a path name, as indicated below:

```
E:\BASIC\FNT0UCH.0BJ
```

See the chapter titled *Understanding MS-DOS Concepts* for more information on file specification and path names.

Input Files

If an input file (object module) specification or a library file specification does not contain a filename extension, then MS-LINK assumes the extensions listed in Table 10-3 below:

Table 10-3. Default Extensions for Object and Library Filenames

Type of File	Default Extension
Object	.OBJ Library .LIB

Output Files

MS-LINK produces the following two types of output files:

- RUN files
- LIST files

MS-LINK appends to the names of these files the extensions listed in Table 10-4 below:

Table 10-4. Default Extensions for Output Filenames

Type of File	Default Extension
Run	.EXE (cannot be changed by the user)
List	.MAP (can be changed by the user)

Virtual Memory File (VM.TMP File)

MS-LINK uses available memory during linking operations. If the files that are to be joined create an output file that exceeds available memory, MS-LINK automatically creates a temporary file and names it "VM.TMP".

In such a case, MS-LINK displays the following message on the screen:

```
VM.TMP has been created
Do not change disc in drive <drv:>
```

If this message appears, do not remove the disc from the indicated drive until linking operations are finished. Otherwise, MS-LINK can behave erratically, and might return the following error message:

```
Unexpected end of file on VM.TMP
```

MS-LINK writes the contents of VM.TMP to the file name that you enter in response to the "Run File:" prompt.

It is important to understand that VM.TMP is not a permanent working file. MS-LINK deletes it when the linking session ends. Therefore, do not assign the name "VM.TMP" to any file that you, the compiler, or the assembler creates.

Caution



If a file that you have named VM.TMP is present on the disc that is in the default drive, and if MS-LINK needs to create a VM.TMP file, then MS-LINK will delete the existing VM.TMP file and will create a new VM.TMP file. Any data in the first VM.TMP file will be lost.

Running MS-LINK

To run MS-LINK, you must perform the following three actions:

- Enter a command to load and start MS-LINK
- Enter filenames in response to the four MS-LINK prompts
- Specify the settings of the optional switches that control other MS-LINK features

Working With MS-LINK

You can run MS-LINK in any one of the following three ways:

- By entering each filename from the keyboard in response to an individual prompt.
- By entering all of the filenames on the same line on which you enter the command that loads and starts MS-LINK.
- By entering the name of a separate Response File, containing the necessary filenames, on the same line with the command that loads and starts MS-LINK.

The syntax for each is shown in Table 10-5 below.

Table 10-5. Summary and Syntax for the Three Ways to Run MS-LINK

Way	Syntax
Text Prompts	LINK
Command-Line	LINK <filename...>[/<switch...>
Response-File	LINK @<file name>



Using the Text-Prompt Method

Using this method, you load MS-LINK into memory by entering the following command:

```
LINK
```

MS-LINK then displays four text prompts on the screen, one at a time. Your responses to these prompts command MS-LINK to perform specific tasks.

These four prompts are discussed in detail in the section below entitled *Command Prompts*.

The optional switches are described in more detail in the section entitled *Switches*.

Using the Command-Line Method

To run MS-LINK using this method, you must enter a statement having the following syntax:

```
LINK <object-list>,[<runfile>],[<listfile>],[  
  <lib-list>][/<switch>...]
```

The entries that follow the word LINK are your responses to the four prompts. As indicated above, the entry fields for the different prompts must be separated by a comma.

In this statement,

object-list	is a list of object modules, separated by a plus-sign or a space
runfile	is the name of the file that will receive the executable output
listfile	is the name of the file that will receive the listing
lib-list	is a list of the library modules to be searched, separated by a blank space
/switch	is one or more optional switches, which you can specify after any of the response entries (either directly before any of the commas, or after <lib-list>, as shown above).

Note

Only object-list is required; the other responses are optional. See the section entitled *The Command Prompts* for more information.

Table 10-6 below lists the default extensions that MS-LINK assumes if you do not specify filename extensions in the first four items in the command line.

Table 10-6. Default Filename Extensions for Responses to Prompts

Name of Response Field	Default Filename Extension
<object-list>	.OBJ
<runfile>	.EXE
<listfile>	.MAP
<lib-list>	.LIB

To select the default extension for a response field, you need only enter a second comma, not preceded by a space, after a first comma, as shown in the following example:

```
LINK DEER+COMET+PRANCER+VIXEN/M/P,  
  DEERLIST,COBLIB.LIB
```

This statement loads MS-LINK, and then loads the following object modules:

```
DEER.OBJ  
COMET.OBJ  
PRANCER.OBJ  
VIXEN.OBJ
```


MS-LINK then does the following things, in the order listed below:

- Links the object modules
- Searches the library file COBLIB.LIB
- Creates a list file named DEERLIST.MAP
- Produces a global symbol map (as instructed by the /M switch)
- Pauses (as instructed by the /P switch)

Then, when you press any key, MS-LINK produces the default DEER.EXE run file. MS-LINK chooses "DEER" for the default name because "DEER" is the name of the first object file in the list. If a semicolon terminates an incomplete command line, default values are assigned, and no prompts are given.

Using the Response-File Method

To call MS-LINK using this method, enter a statement having the following syntax:

```
LINK @<file name>
```

In this statement,

@ is the character reserved by MS-LINK to indicate that the filename that follows is a response file

file name is the name of a response file

A response file contains answers to the MS-LINK prompts, and can also specify one or more of the switches. This

method lets you conduct the MS-LINK session without having to enter responses to the MS-LINK prompts one at a time.

However, before using this method to invoke MS-LINK, you must create the Response File; if you do not do this, MS-LINK will display the following error message:

```
"[<file name>]"  
Cannot open response file.
```

When you use this method, MS-LINK displays each prompt in turn, followed by the response(s) from the response file. If the response file does not contain one or more answers to each prompt (in the form of either a filename, a command character, or

Creating a Response File

You can use any text editor, such as EDLIN, or the MS-DOS COPY command to create a response file. The file contains four text lines (one for each of the MS-LINK prompts). The responses must appear in the same order in which the command prompts appear, and must be separated from the previous response by a carriage return. You can use alone on a line to pass over a prompt and use the default value. If no default has been set for that prompt, LINK will wait for you to enter an appropriate response from the keyboard before continuing.

The following is an example of a response file:

```
DEER COMET PRANCER VIXEN  
/PAUSE/MAP  
DEERLIST  
C\BLIB.LIB
```

The first line of this response file tells MS-LINK to load the following four object modules:

```
DEER.OBJ
COMET.OBJ
PRANCER.OBJ
VIXEN.OBJ
```

The second line instructs MS-LINK to pause (to let you swap discs) before generating the .EXE file which defaults to DEER.EXE. (You may wish to review the section below entitled *Switches* before using the /PAUSE switch. However, in brief, this switch halts linking operations just before MS-LINK writes the .EXE file. You must press the key to resume linking operations.)

In response to the third line, MS-LINK assigns the name DEERLIST.MAP to the output files.

Last, as instructed in the fourth line, MS-LINK will search the library file COBLIB.LIB.

You can use the command characters "+" and ";" and the switches in a response file the same way you use them in entries from the keyboard. See the sections on *Command Characters* and *Optional Switches* for more information. The following is an example of using command characters in a response file:

```
DEER+COMET+PRANCER+VIXEN
/PAUSE/MAP
DEERLIST;
```

Command Prompts

MS-LINK executes the commands that you enter in response to the following four text prompts:

- Object Modules [.OBJ]
- Run File [First-Object-filename.EXE]
- List File [NUL.MAP]
- Libraries [.LIB]

After you enter a response to one prompt, the next prompt appears. After you enter a response to the last prompt, MS-LINK automatically starts linking the object files. At this point, you do not need to enter any further commands.

Once the link session has ended, MS-LINK returns control to MS-DOS. The re-appearance of the MS-DOS prompt (for example, B>) on the screen indicates that the MS-LINK session was successful. If the link session did not end successfully, the appropriate MS-LINK error message appears on the screen.

MS-LINK prompts you for the names of the object, run, and list files, and for the name of the library file(s). If a prompt has a default response, that default response appears in brackets after the prompt. (The "Object Modules:" prompt is followed only by a default filename-extension response because this prompt has no default filename response; instead, it requires that you enter a filename.)

Table 10-7 below lists the four prompts and the corresponding actions that you can take.

Table 10-7. The MS-LINK Command Prompts and User Responses

Prompt	Response to be Made by User
Object Modules [.OBJ]	Enter name(s) of .OBJ file(s) to be linked. (Default: none. You must enter a response)
Run File [Object-file.EXE]:	Enter filename for executable object code. (Default: First-Objectfilename.EXE)
List File [NUL.MAP]:	Enter filename for output listing. (Default: NUL filename; no list file will be created)
Libraries [.LIB]:	Enter names of files to be searched. (Default: MS-LINK makes no search)

At the end of each response line, you can specify one or more optional switches. Each switch must be preceded by a slash (/). Switches are discussed in detail in the section on *Optional Switches*.

Object Modules [.OBJ]:

You must enter a list of the object modules to be linked. MS-LINK assumes that the filename extension is .OBJ unless you specify a different extension in response to this prompt.

Modules must be separated by a plus-sign (+) or a space.

Recall that MS-LINK loads segments into classes in the order in which it meets them; this condition is useful when you are deciding on the order in which to enter the object modules.

Run File [First-Object-filename.EXE]:

MS-LINK assigns the specified filename to the RUN (executable) file that it creates to contain the results of the linking session. MS-LINK assigns the filename extension ".EXE" to all RUN files, ignoring any other extension that you might have specified.

If you do not enter a response to the Run File: prompt, then MS-LINK takes the first filename entered in response to the Object Modules: prompt and uses it as the name of the RUN file. MS-LINK assumes that you want the .EXE file placed in the current directory, even if your response to the Object Modules: prompt specifies another path or drive.

List File [NUL.MAP]:

Your response to this prompt instructs MS-LINK to generate a listing file that contains an entry for each segment in the input (object) modules. Each of these entries also shows the offset (addressing) in the RUN file.

A typical list file might look like this:

Start	Stop	Length	Name	Class
00000H	0002CH	002DH	CODE	CODE
0002DH	00D74H	0D48H	XCODE	MORECODE
00D75H	00DA2H	002EH	DSEG	DATA
00DBOH	00FAFH	0200H	STACK	STACK
00fBOH	010E5H	0136H	VECTOR	VECTORS

Origin	Group
00D7:0	DGROUP
0000:0	PGROUP

If this prompt is passed over, the default NUL.MAP is specified, and MS-LINK does not generate a listing file.

Libraries [.LIB]:

A valid response to this prompt contains from one through eight library filenames, or else press alone to indicate that you do not want MS-LINK to search any library files.

The library files must have been created beforehand by a library utility such as the MS-DOS LIB utility documented in the *Vectra MS-DOS Macro Assembler* manual, or created as part of a pre-packaged library, such as the library available with a highlevel language. MS-LINK assumes that each library file has the filename extension ".LIB".

Library filenames must be separated by a blank space or by a plus sign (+).

MS-LINK searches the library files, in the order in which the files were entered, to resolve external references. When it finds the module that defines the external symbol,

MS-LINK processes that module as though it were another object module.

If MS-LINK cannot find the library file on the discs that are currently in the disc drives, the following message appears on the screen:

```
Cannot find library <library-name>  
Enter new drive letter:
```

In response, you need only type the letter designating the desired drive (for example, "B"), without entering a colon after the letter.

To find references, MS-LINK uses a method known as a "dictionary-indexed library search." Under this approach, MS-LINK finds definitions for external references via access to an index, instead of searching for each reference by scanning a file sequentially from beginning to end. This method saves time during a linking session that requires a library search.

When using object modules generated by a compiler, you should know that the compiler may assume the existence of certain default library files. For example, MS-COBOL (Version 1.10) assumes that the library files COBOL1.LIB and COBOL2.LIB exist. In this case, if you are using MS-LINK to join COBOL object (.OBJ) modules, you should enter the names of these library files.

Command Characters

MS-LINK provides the following three command characters:

- The plus-sign (+)
- The semicolon (;)
- CTRL-C

The Plus-Sign

You can use a plus-sign (+) to separate entries and to enter a series of items in your response to the `Object Modules` prompt and to the `Libraries` prompt. (You can also use a blank space, or a mixture of blank spaces and plus-signs, to separate object modules on the same line.)

To enter a series of responses that occupy more than one line (80 characters, less the number of characters occupied by the prompt), enter a plus-sign followed by `[Enter]` at the end of the line. If the Plus-sign/`[Enter]` is the last entry in response to the `Object Modules` or `Libraries` prompt, then MS-LINK will display the `Object Modules` or `Libraries` prompt again, and you can continue to enter responses.

When you have entered the names of all of the modules to be linked, be sure that the response line ends with a module name and `[Enter]` — not with a Plus-sign/`[Enter]`.

For example, to use spaces and plus-signs to enter a set of modules named SLEEPY, DOPEY, HAPPY, FRODO, and DOC, you can enter the following responses in response to the `Object Modules` prompt:

```
Object Modules [.OBJ]: SLEEPY DOPEY+[Enter]
Object Modules [.OBJ]: HAPPY+FRODO+[Enter]
Object Modules [.OBJ]: DOC [Enter]
```

The Semicolon

Any time after you respond to the `Object Modules` prompt, you can use a single semicolon (;) followed immediately by `Enter` to select the default responses to the remaining prompts. Entering this command saves time and avoids the need to enter a series of `Enter` keystrokes.

However, once you enter a semicolon, you cannot enter a response to any of the remaining prompts in the current MS-LINK session. Therefore, do not use the semicolon to skip over one or more prompts on your way to a prompt that you do want to respond to; instead, use one or more `Enter` keystrokes to do so.

For example, consider the prompts and responses shown below:

```
Object Modules [·OBJ]: SLEEPY GRUMPY Enter
Run Module [SLEEPY·EXE]: ; Enter
```

Because a semicolon appears directly before the `Enter` in the response to the `Run Module` prompt, none of the subsequent prompts will appear, and MS-LINK will assume the default responses to them (including NUL.MAP for the listing file).

Control-C

To end an MS-LINK session at any time, press `CTRL` and C simultaneously. If you enter an incorrect response (such as an undesired or misspelled filename), you must end the MS-LINK session by entering `CTRL` C, and then begin a new MS-LINK session. `CTRL` - `Break` is equivalent to `CTRL` C.

However, if you have typed an erroneous response but have not entered it (that is, if the cursor is still on the line that contains the mistake), you can press the `←` key to delete each erroneous character, and then type a correct response.

Optional Switches

Table 10-8 below summarizes the optional switch functions.

Table 10-8. The MS-LINK Optional Switch Functions

Switch	Function
/DSALLOCATE	Loads data at the high end of the data segment. Required for PASCAL and FORTRAN programs.
/HIGH	Places the RUN file as high as possible in memory. Must not be used with PASCAL or FORTRAN programs.
/LINENUMBERS	Includes line numbers in the LIST file.
/MAP	Lists all global symbols and their definitions in the .MAP file.
/PAUSE	Pauses the MS-LINK session; causes system to wait until the user presses the <input type="text" value="Enter"/> key.
/STACK: <number>	Sets a fixed stack size in the RUN file.

You must specify optional switches at the end of a prompt response, regardless of the method used to invoke MS-LINK. You can group the switch specifications at the end of any one response, or you can place them at the end of several responses. If you include more than one switch at the end of one response, each switch must be preceded by a slash (/).

For example:

```
Object Modules [.OBJ]:SNEEZY/DS/HI/STACK:512
```

The switch specifications can be spelled out in whole, or can be truncated. However, a truncation must consist of a sequential subset of the letters in the specification, starting with the first letter. No letter(s) can be skipped or transposed. The examples listed below show examples of legitimate and unacceptable truncations for a switch specification; the /DSALLOCATE switch has been used as a sample:

Valid	Invalid
/D	/DSL
/DS	/DAL
/DSA	/DL
/DSALLO	/ALLOC
/DSALLOCA	/DSALLOCT

The /DSALLOCATE Switch

This switch instructs MS-LINK to load all data in the group (DGROUP) at the high end of the group. (Otherwise, MS-LINK loads all data at the low end of the group.)

At runtime, the DS pointer is set to the lowest possible address, thus letting the entire data segment be used. If the /DSALLOCATE switch is used with the /HIGH switch, the user application can dynamically allocate any available memory located below the area specifically allocated within DGROUP. However, this dynamically allocated memory can still be addressed by the same DS pointer. Certain compilers need this type of dynamic allocation.

Note

Your application program can dynamically allocate up to 64K bytes, less the amount allocated within DGROUP.

The /HIGH Switch

The /HIGH switch instructs MS-LINK to place the image of the RUN file as high as possible in memory. Otherwise, MS-LINK places the image of the RUN file as low as possible in memory. This action can cause the program to be written over the transient portion of PAM or COMMAND.COM file. In such a case, MS-DOS displays one of the following error messages:

```
Insert COMMAND.COM in default drive and strike  
any key when ready
```

or

```
Insert System Disc in A: and  
strike any key to continue.
```

Note

Do not use the /HIGH switch with PASCAL or FORTRAN programs. They will not run.

The /LINENUMBERS Switch

The /LINENUMBERS switch instructs MS-LINK to include in the listing file the line numbers and addresses of the source statements in the input modules. (Otherwise, MS-LINK omits line numbers from the listing file.)

Not all compilers produce object modules that contain line-number information. If the object module doesn't contain line-number information, then MS-LINK cannot include it.

The /MAP Switch

The /MAP switch instructs MS-LINK to list all public (global) symbols defined in the input modules. If you do not specify /MAP, then MS-LINK will list only errors (including undefined global symbols), symbols, classes, and groups.

The MAP switch produces two lists: one in alphabetical order and one in value order. The lists state the value of each symbol, and also gives its segment:offset location in the RUN file.

After MS-LINK locates and searches the specified library file(s) (if told to do so), it produces a map listing the segments in the order in which they appear in the RUN (.EXE) output file. A typical linker map appears in Figure 10-3 below:

a)	Start	Stop	Length	Name
	00000H	009ECH	09EDH	CODE
	009F0H	01166H	0777H	SYSINITSEG

Figure 10-3. Contents of a Typical MS-LINK Linker Map, continued on next page

b)	Address	Publics by Name
	009F:0012	BUFFERS
	009F:0005	CURRENT__DOS__LOCATION
	009F:0011	DEFAULT__DRIVE
	009F:000B	DEVICE__LIST
	009F:0013	FINAL__DOS__LOCATION
	009F:000F	MEMORY__SIZE
	009F:0000	SYSINIT
	Address	Publics by Value
	009F:0000	SYSINIT
	009F:0005	CURRENT__DOS__LOCATION
	009F:0009	FINAL__DOS__LOCATION
	009F:000B	DEVICE__LIST
	009F:000F	MEMORY__SIZE
	009F:0011	DEFAULT__DRIVE
	009F:0012	BUFFERS

Figure 10-3. Contents of a Typical MS-LINK Linker Map, continued

In part (a) of this example, the "Start" and "Stop" columns do not contain the absolute addresses where these segments are loaded. Instead, they contain the 20-bit hex address of each segment relative to the beginning of the load module. (This point is known as "location zero.")

In part (b) of this example, MS-LINK lists the public symbols: first alphabetically by name, and then by value.

The /PAUSE Switch

The /PAUSE switch causes MS-LINK to pause the linking operations until you press the `Enter` key. (Otherwise, MS-LINK runs through the linking session non-stop from beginning to end.) The pause lets you change discs before MS-LINK outputs the RUN (.EXE) file.

When MS-LINK encounters the /PAUSE switch, it displays the following message:

```
About to generate .EXE file
Change disc press Enter
```

However, do not remove a disc that is to receive the List file, or a disc that contains the VM.TMP file (that is, if MS-LINK created a VM.TMP file).

The /STACK:<number> Switch

In this specification, <number> represents any positive integral numerical value (expressed in decimal notation) up to 65,536 bytes. If you specify a value from 1 through 511, then MS-LINK uses 512. If you do not specify the /STACK switch, then MS-LINK looks for stack size information in the .OBJ files.

All compilers and assemblers should provide information in the object modules that lets MS-LINK compute the required stack size. At least one object module must contain a stack-allocation statement. Otherwise, MS-LINK will display the following error message:

```
Warning: No Stack Statement
```


Error Messages

If an error occurs during an MS-LINK linking session, the linker reports the error and attempts to continue the linking session. Therefore, after you determine the cause and correct the erroneous condition, you must restart MS-LINK and conduct the linking session again.

The MS-LINK error messages are listed below, along with explanations and solutions or suggested actions.

Message:	ATTEMPT TO ACCESS DATA OUTSIDE OF SEGMENT BOUNDS, POSSIBLY BAD OBJECT MODULE
Cause:	An object file is deficient somehow.
Remedy:	Recompile object modules.
Message:	BAD NUMERIC PARAMETER
Cause:	A numerical value is not expressed in digits.
Remedy:	Express the numerical value in digits.
Message:	CANNOT OPEN TEMPORARY FILE
Cause:	MS-LINK cannot create the VM.TMP virtualmemory file because the disc directory is full.
Remedy:	Insert a new disc. However, do not change the disc that is to receive the LIST.MAP file; in this case, restart LINK.
Message:	ERROR: DUP RECORD TOO COMPLEX
Cause:	The DUP record in an assembly-language module is too complex.

Remedy: Simplify DUP record in assembly language program.

Message: ERROR: FIXUP OFFSET EXCEEDS
FIELD WIDTH

Cause: An assembly-language instruction refers to an address with a short instruction instead of with a long module.

Remedy: Edit the assembly-language source and re-assemble the instruction.

Message: INPUT FILE READ ERROR



Cause: An object file is deficient somehow.

Remedy: Recompile the file and try again.

Message: INVALID OBJECT MODULE

Cause: One or more object modules were improperly formed, or were incomplete (for example, as would be the case if assembly were stopped in mid-process).

Remedy: Re-compile the object module(s).

Message: SYMBOL DEFINED MORE THAN ONCE

Cause: MS-LINK found two or more modules that define a single symbol name.

Remedy: Remove the definition or change the name in one of the conflicting modules. You should be aware that some languages use only the first eight characters in a symbol name.

Message: PROGRAM SIZE OR NUMBER OF SEGMENTS
EXCEEDS CAPACITY OF LINKER

Cause:	The aggregate size of all of the files, once linked, cannot exceed 384K bytes; the total number of segments cannot exceed 255.
Remedy:	Review the size of the files and the number of segments; reduce the aggregate size by combining segments.
Message:	REQUESTED STACK SIZE EXCEEDS 64K
Cause:	A stack size greater than 64K bytes was requested.
Remedy:	Use the /STACK switch to specify a stack size that is equal to or less than 64K bytes.
Message:	SEGMENT SIZE EXCEEDS 64K
Cause:	64K bytes is the upper limits of the addressing system.
Remedy:	Reduce segment size by splitting it into two or more segments.
Message:	SYMBOL TABLE CAPACITY EXCEEDED
Cause:	Many long names were entered (occupying approximately 25K bytes).
Remedy:	Review the number and length of the names entered in the symbol table and shorten names.
Message:	TOO MANY EXTERNAL SYMBOLS IN ONE MODULE
Cause:	The upper limit on external symbols per module is 256.

Remedy: Reduce the number of external symbols per module by eliminating names or by splitting one or more modules.

Message: TOO MANY GROUPS

Cause: The upper limit on groups is 10.

Remedy: Recommended Action: Declare fewer groups.

Message: TOO MANY LIBRARIES SPECIFIED

Cause: The upper limit on library files is 8.

Remedy: Reduce the number of library files specified in response to the "Libraries" prompt. Use the Microsoft LIB Utility available and documented in the Programmer's Tools, to combine libraries, and/or remove unnecessary routines.

Message: TOO MANY PUBLIC SYMBOLS

Cause: The upper limit on public symbols is 1024.

Remedy: Declare fewer public symbols.

Message: TOO MANY SEGMENTS OR CLASSES

Cause: The upper limit on segments and classes (considered together) is 256.

Remedy: Declare fewer segments or classes.

Message: UNRESOLVED EXTERNALS: <list>

Cause: For the external symbols listed, MS-LINK found no defining module among the modules or library files specified.

Remedy: Include the necessary definition module among the specified modules or library files by placing file name after Object Module or Library prompt.

Message: VM READ ERROR

Cause: A disc problem, not caused by MS-LINK.

Remedy: Check your system

Message: Warning : NO STACK SEGMENT

Cause: None of the object modules specified contains a statement allocating stack space.

Remedy: Review the object modules; at least one object module should contain a stack-allocation statement.

Message: Warning : SEGMENT OF ABSOLUTE OR UNKNOWN TYPE

Cause: An object module is bad, or an improper module was loaded for linking (i.e., a module MS-LINK cannot handle, such as an absolute object module).

Remedy: Verify the type of module and/or segment loaded.

Message: WRITE ERROR IN TMP FILE

Cause: No more disc space remains into which the VM.TMP file can expand.

Remedy: Free some disc space by moving or deleting files.

Message: WRITE ERROR ON RUN FILE

Cause: The remaining disc space is too small for the RUN file.

Remedy: Free some disc space by moving or deleting files.

MS-LINK Examples

The following examples demonstrate the use of the MS-LINK Utility.

Example 1:

This example demonstrates the linking of object modules from several discs and several directories, and the use of the "+" command character to enter a series of responses to the Object Modules prompt. Note, too, the use of the semicolon (;) to select default filenames for the Run File and List File, to and pass over the Library search.

```
B>link
Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983
Object Modules [.OBJ] : MAIN+SECOND+BIGFILE+
Object Modules [.OBJ] : E:\BASIC\FNTOUCH+
Object Modules [.OBJ] : C:CHAR
Run File [MAIN.EXE] : ;
```

```
B>
```

Example 2:

In this example, the user has supplied the name of the Runfile as RUN_FILE. Otherwise, the Linker would have used the name of the first Object Module and named the Run File MAIN.EXE. Note use of the semicolon to skip over the remaining prompts.

```
B>LINK
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
Object Modules [.OBJ] : MAIN+SECOND  
Run File [MAIN.EXE] : RUN_FILE;
```

```
B>
```


Example 3:

In the third example, the user specified a List File name, then used the MS-DOS TYPE command to view its contents. Note use of the semicolon to skip over the remaining prompts.

```
B>LINK
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
Object Modules [.OBJ] : MAIN SECOND BIGFILE  
Run File [MAIN.EXE] :  
List File [NUL.MAP] : MAIN;
```

```
B>TYPE MAIN.MAP
```

Start	Stop	Length	Name	Class
00000H	0002CH	002DH	CODE	CODE
0002DH	00074H	0048H	XCODE	MORE_CODE
00075H	000A2H	002EH	DSEG	DATA
000B0H	00FAFH	0200H	STACK	STACK
00FB0H	010E5H	0136H	VECTOR	VECTORS

Origin	Group
0007:0	DGROUP
0000:0	PGROUP

```
B>
```

Example 4:

Here, the user specified one Object File, skipped over the Run File and List file prompts causing default file names to be used, and specified a Library Search by entering a file name in response to the Libraries prompt.

```
B>LINK
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
Object Modules [.OBJ] : MAIN  
Run File [MAIN.EXE] :  
List File [NUL.MAP] :  
Libraries [.LIB] : LIBRARY
```

```
B>
```

Example 5:

Example 5 is the same as Example 1, except that the user has placed all entries specifying Object Modules on the command line (the Command-Line Method), separated with the "+" command character:

```
B>LINK  
MAIN+SECOND+BIGFILE+E:\BASIC\FNTOUCH+C:CHAR;  
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
B>
```

Example 6:

This example achieves the same result as in Example 3, but the user has placed all instructions on the command line rather than responding to individual prompts:

```
B>LINK MAIN SECOND BIGFILE,,MAIN;

Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983

B>TYPE MAIN.MAP

Start      Stop      Length   Name      Class
00000H     0002CH   002DH    CODE      CODE
0002DH     00D74H   0D48H    XCODE     MORE_CODE
00D75H     00DA2H   002EH    DSEG      DATA
00DB0H     00FAFH   0200H    STACK     STACK
00FB0H     010E5H   0136H    VECTOR    VECTORS

Origin     Group
00D7:0     DGROUP
0000:0     PGROUP

B>
```

Note the following:

- Both “+” and space were used as separators on the command line
- One comma followed by a second comma was used to skip the Run File prompt.
- The semicolon was used to skip the Library prompt so that the no library search was made.

Finally, the user used the MS-DOS TYPE command to view the contents of the List file MAIN.MAP.

Example 7:

In the example that follows, the user has created a response file named LINKOBS and used the MS-DOS TYPE command to view the contents. The command to call and run MS-LINK is followed by

```
LINK @LINKOBS
```

enabling the response file to respond to the four prompts.

```
B>TYPE LINKOBS  
MAIN+SECOND+BIGFILE+  
E:\BASIC\FNTOUCH+ C:CHAR ;
```

```
B>LINK @LINKOBS
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
Object Modules [.OBJ] : MAIN+SECOND+BIGFILE+  
Object Modules [.OBJ] : E:\BASIC\FNTOUCH+  
Object Modules [.OBJ] : C:CHAR  
Run File [MAIN.EXE] ;
```

```
B>
```

If the Linking session had not been successful, error messages would have been displayed before control was returned to MS-DOS.

Example 8:

In this example, the response file method is again used. The Linker was unable to locate the Object Module LONGFILENAME.OBJ and prompted the user to change discs; the user terminated the session with the CTRL C command character.

```
B>TYPE INVALID
LONGFILENAME
LONG
```



```
B>LINK @INVALID
```

```
Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
Object Modules [.OBJ] : LONGFILENAME
Run File [LONGFILENAME.EXE] : LONG
List File [NUL.MAP] :
Libraries [.LIB] :
Cannot find file LONGFILENAME.OBJ
change diskette <hit ENTER> ^C
```

```
B>
```

Example 9:

In this example, the user failed to specify a Library File and the Link session paused. The user entered the library file so the session could continue.

```
B>TYPE THREE.PAR
main second bigfile
mainprog
mainmap
```

```
B>link @three.par
Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
Object Modules [.OBJ] : MAIN SECOND BIGFILE
Run File [MAIN.EXE] : MAINPROG
List File [NUL.MAP] : MAINMAP
Libraries [.LIB] : library
```

```
B>
```

Example 10:

The following lines of code illustrate the use and non-use of the /DS switch. In the first part of the sample code, the /DS switch is used causing data to be placed in high memory. In the second part of the sample code, the switch is not used, and data is placed in low memory.

```
B>LINK
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
Object Modules [.OBJ] : SAMPLE/DS  
Run File [SAMPLE.EXE] :  
List File [NUL.MAP] : CON;
```

Start	Stop	Length	Name	Class
00000H	00012H	0013H	YCODE	CODE
00020H	0003AH	001BH	CODE	CODE
0003BH	00051H	0017H	DSEG	DATA
00052H	00D99H	0D48H	XCODE	MORE_CODE
00DA0H	00ED5H	0136H	VECTOR	VECTORS
00EE0H	010DFH	0200H	STACKSE	STACK

```
Origin Group  
FOEE:0 DGROUP  
0002:0 PGROUP
```

```
Program entry point at 0002:0000
```

```
B>LINK
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983  
Object Modules [.OBJ] : SAMPLE  
Run File [SAMPLE.EXE] :  
List File [NUL.MAP] : CON;
```


Start	Stop	Length	Name	Class
00000H	00012H	0013H	YCODE	CODE
00020H	0003AH	001BH	CODE	CODE
0003BH	00051H	0017H	DSEG	DATA
00052H	00099H	0048H	XCODE	MORE_CODE
000A0H	000D5H	0136H	VECTOR	VECTORS
000E0H	010DFH	0200H	STACK_SEG	STACK

Origin	Group
0003:0	DGROUP
0002:0	PGROUP

Program entry point at 0002:0000

B>

Example 11:

Because the /HIGH switch was used in this example, COMMAND.COM was overwritten in memory. The Operating System notified the user with an error message instructing him to insert the disc with the COMMAND.COM file and start again. In the second try, the user omitted the /HIGH switch and the Linking session was successful.

```
B>LINK SAMPLE,A:SAMPLE/HIGH;
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
B>A:
```

```
A>SAMPLE Simple linker example
```

```
Insert COMMAND.COM disk in default drive and  
strike any key when ready
```

```
A>B:
```

```
B>LINK SAMPLE,A:SAMPLE;
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
B>A:
```

```
A>SAMPLE Simple linker example
```

```
A>B:
```

```
B>
```

Example 12:

This example illustrates the use and non-use of the /MAP switch. In the first part of the sample code, the /MAP switch is used; in the second part of the sample, it is not. Note, too, that list output was directed to the console.

```
B>LINK SAMPLE,,,CON/MAP;
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

Start	Stop	Length	Name	Class
00000H	00012H	0013H	YCODE	CODE
00020H	0003AH	001BH	CODE	CODE
0003BH	00051H	0017H	DSEG	DATA
00052H	00099H	0048H	XCODE	MORE_CODE
000A0H	000D5H	0136H	VECTOR	VECTORS
00EE0H	010DFH	0200H	STACK_SEG	STACK

Origin	Group
0003:0	DGROUP
0002:0	PGROUP

Address Publics by Value

0000:0000	PRINT_SETUP
0002:0007	PRINT_IT
0005:0002	CALLO
0005:089A	CALLJ
0003:0D70	ARRAY0
0003:0E38	DIRECTION

Program entry point at 0002:0000

B>LINK SAMPLE,,,CON;

Microsoft 8086 Object Linker

Version 3.00 (C) Copyright Microsoft Corp 1983

Start	Stop	Length	Name	Class
00000H	00012H	0013H	YCODE	CODE
00020H	0003AH	001BH	CODE	CODE
0003BH	00051H	0017H	DSEG	DATA
00052H	00099H	0048H	XCODE	MORE_CODE
000A0H	000E5H	0136H	VECTOR	VECTORS
00EE0H	010DFH	0200H	STACK_SEG	STACK

Origin	Group	0003:0	DGROUP
0002:0	PGROUP		

Address Publics by Value

0000:0000	PRINTSETUP
0002:0007	PRINTIT
0005:0002	CALLO
0005:089A	CALLJ
0003:0D70	ARRAY0
0003:0E38	DIRECTION

Program entry point at 0002:0000

B>

Example 13:

This example demonstrates use of the /PAUSE Switch.

```
B>LINK SAMPLE/P;
```

```
Microsoft 8086 Object Linker
```

```
Version 3.00 (C) Copyright Microsoft Corp 1983
```

```
About to generate .EXE file
```

```
Change disks <hit ENTER> 
```

```
B>
```

Example 14:

In this final example, use of the /STACK switch is shown. In the first part of the sample code, the Stack size is specified as 1024 bytes; in the second part of the sample code, the Stack size defaults to 512 bytes.

```
B>LINK SAMPLE/STACK:1024,,CON;
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

Start	Stop	Length	Name	Class
00000H	00012H	0013H	YCODE	CODE
00020H	0003AH	001BH	CODE	CODE
0003BH	00051H	0017H	DSEG	DATA
00052H	00D99H	0D48H	XCODE	MORE_CODE
00DA0H	00ED5H	0136H	VECTOR	VECTORS
00EE0H	012DFH	0400H	STACK_SEG	STACK

```
Origin Group  
0003:0 DGROUP  
0002:0 PGROUP
```

```
Program entry point at 0002:0000
```

```
B>LINK SAMPLE,,CON;
```

```
Microsoft 8086 Object Linker  
Version 3.00 (C) Copyright Microsoft Corp 1983
```

Start	Stop	Length	Name	Class
00000H	00012H	0013H	YCODE	CODE
00020H	0003AH	001BH	CODE	CODE
0003BH	00051H	0017H	DSEG	DATA
00052H	00D99H	0D48H	XCODE	MORE_CODE
00DA0H	00ED5H	0136H	VECTOR	VECTORS
00EE0H	010DFH	0200H	STACK_SEG	STACK

```
Origin  Group
0003:0  DGROUP
0002:0  PGROUP
```

```
Program entry point at 0002:0000
```

```
B>
```



No.	Name	Grade
1	John A. Smith	H
2	James B. Jones	H
3	Robert C. Brown	H
4	William D. White	H
5	Thomas E. Black	H
6	Charles F. Green	H
7	Richard G. Gray	H
8	Henry H. Blue	H
9	George I. Yellow	H
10	Frank J. Purple	H
11	Edward K. Red	H
12	George L. Orange	H
13	Harold M. Pink	H
14	Arthur N. Brown	H
15	Donald O. Green	H
16	Paul P. White	H
17	Samuel Q. Black	H
18	John R. Yellow	H
19	Robert S. Purple	H
20	Richard T. Red	H
21	William U. Orange	H
22	Thomas V. Pink	H
23	Charles W. Brown	H
24	George X. Green	H
25	Harold Y. White	H
26	Arthur Z. Black	H
27	Edward AA. Yellow	H
28	Frank BB. Purple	H
29	George CC. Red	H
30	Harold DD. Orange	H
31	Arthur EE. Pink	H
32	Donald FF. Brown	H
33	Paul GG. Green	H
34	Samuel HH. White	H
35	John II. Black	H
36	Robert JJ. Yellow	H
37	Richard KK. Purple	H
38	William LL. Red	H
39	Thomas MM. Orange	H
40	Charles NN. Pink	H
41	George OO. Brown	H
42	Harold PP. Green	H
43	Arthur QQ. White	H
44	Edward RR. Black	H
45	Frank SS. Yellow	H
46	George TT. Purple	H
47	Harold UU. Red	H
48	Arthur VV. Orange	H
49	Donald WW. Pink	H
50	Paul XX. Brown	H
51	Samuel YY. Green	H
52	John ZZ. White	H
53	Robert AAA. Black	H
54	Richard BBB. Yellow	H
55	William CCC. Purple	H
56	Thomas DDD. Red	H
57	Charles EEE. Orange	H
58	George FFF. Pink	H
59	Harold GGG. Brown	H
60	Arthur HHH. Green	H
61	Edward III. White	H
62	Frank IIII. Black	H
63	George IVV. Yellow	H
64	Harold VVV. Purple	H
65	Arthur VII. Red	H
66	Donald VIII. Orange	H
67	Paul IX. Pink	H
68	Samuel X. Brown	H
69	John XI. Green	H
70	Robert XII. White	H
71	Richard XIII. Black	H
72	William XIV. Yellow	H
73	Thomas XV. Purple	H
74	Charles XVI. Red	H
75	George XVII. Orange	H
76	Harold XVIII. Pink	H
77	Arthur XIX. Brown	H
78	Edward XX. Green	H
79	Frank XXI. White	H
80	George XXII. Black	H
81	Harold XXIII. Yellow	H
82	Arthur XXIV. Purple	H
83	Donald XXV. Red	H
84	Paul XXVI. Orange	H
85	Samuel XXVII. Pink	H
86	John XXVIII. Brown	H
87	Robert XXIX. Green	H
88	Richard XXX. White	H
89	William XXXI. Black	H
90	Thomas XXXII. Yellow	H
91	Charles XXXIII. Purple	H
92	George XXXIV. Red	H
93	Harold XXXV. Orange	H
94	Arthur XXXVI. Pink	H
95	Edward XXXVII. Brown	H
96	Frank XXXVIII. Green	H
97	George XXXIX. White	H
98	Harold XL. Black	H
99	Arthur XLI. Yellow	H
100	Donald XLII. Purple	H

11

EDLIN

Introduction to EDLIN

EDLIN, the MS-DOS line editor, can be used to create, change, and display source files and text files. Source files are defined as files containing program listings; text files are defined as files containing data in legible format, such as an unformatted word processing file. Text files will be used as examples in this chapter.

In addition to allowing you to create new files and save them on a disc, EDLIN allows you to:

- Update existing files and save both the original and updated versions.
- Delete, edit, insert, and display lines in a file.
- Search for, delete, or replace text within one or more lines in a file.

The files you create or edit using EDLIN are divided into lines, and each line may be up to 253 characters in length. Line numbers are generated and displayed during the editing process, but are not present in the saved file.

Lines are always numbered consecutively in EDLIN files, regardless of the editing functions performed. For example, when you insert lines, all line numbers following inserted text are incremented by the number of lines inserted; and when you delete lines in a file, the line numbers following the deleted text are decremented by the number of lines deleted.

How To Start EDLIN

To start EDLIN, enter:

```
EDLIN [path] >file name< [/B]
```

path	specifies the path to the file name you wish to create or edit.
file name	specifies the file you wish to create or edit.
/B	instructs EDLIN to ignore any embedded end-of-file marks (CTRL Z) in your file and process the entire file. This is useful for editing files which are known to contain embedded end-of-file marks.

When you access EDLIN, you will see the following display:

```
B<EDLIN EXAMPLE
New file
*
```

If EDLIN does not find the specified file on the active drive or the specified drive, a new file will be created using the file name you enter. EDLIN responds with the words **New file** to indicate that a new file is being created.

In this example, the pathname included only a file name; when data has been entered into this file, and the EDLIN session is ended, it will be saved to the current directory on the disc in the default drive. The asterisk (*) is the EDLIN prompt.

If you specify an existing file, EDLIN will attempt to load it into memory; EDLIN loads lines until memory is 75% full to allow room to add lines during editing, and displays the * prompt. If your file is too large to load, you will have to save some of the edited lines to disc during the editing process in order to free memory so you can load the remaining lines for editing. The WRITE and APPEND commands spell out the procedures for doing this. To begin entering text, enter an I (Insert) command to insert lines into EXAMPLE.

If you do not specify a filename extension when you specify a new file, one will not be assigned by EDLIN; when you edit an existing file, however, EDLIN will assign an extension of .BAK to your original file (to the file on disc) and your edited version will have the filename specified when you accessed EDLIN; in this way, EDLIN automatically keeps a backup copy of each file you create. The .BAK file cannot be edited. If you wish to edit it, you must first rename it with a different extension, then start EDLIN.

When you have completed your editing session, you leave EDLIN and save your file to disc using the END command.

Caution

When you create or edit an EDLIN file, be certain there is enough space on the specified disc to save the file; if there is not enough space, you risk losing part of your data when you attempt to save it to disc.

Information Common To All EDLIN Commands

EDLIN commands perform editing functions on lines of text. The following information is common to all EDLIN commands, and you should be familiar with it before you create or edit an EDLIN file:

- Pathnames are acceptable as options to commands. For example, typing:

```
EDLIN BIN\USER\JOE\TEXT.TXT
```

allows you to edit the TEXT.TXT file in the subdirectory JOE.

- With the exception of the EDIT command, all commands consist of a single letter.
- With the exception of the END and QUIT commands, commands may be preceded and/or followed by parameters.
- Commands and string parameters may be entered in either upper- or lower-case letters, or a combination of both.
- Commands and parameters may be separated by delimiters for readability; however, a delimiter is only required between two adjacent line numbers. Delimiters are spaces or commas.

For example, to delete line 6, the command `6D` is the same as the command `6, D`.

- You can refer to line numbers relative to the current line in your EDLIN file. Use a minus sign (-) followed by a number to indicate a line preceding the current line. Use a plus sign (+) followed by a number to indicate a line following the current line. For example:

```
-10,+10 L
```

displays 10 lines preceding the current line, the current line, and 10 lines following the current line. Note the use of the comma between adjacent line numbers.

- You can enter multiple commands on one line using an appropriate separator. When you enter an Edit command (<line>), subsequent commands must be separated from <line> by a semicolon. For all other commands, one command may follow another separated by either a comma or a space. In the case of a Search or Replace command, however, <string> must be ended by **CTRL** Z instead of **Enter**.



Example

The following command allows you to edit line 15, then list lines 10 through 20:

```
15;-5,+5L
```

In the next example, the command instructs EDLIN to search for "This string", then display the preceding five lines and the following five lines containing the matched string. If no match is found for the search string, the lines listed will be those lines relative to the current line.

```
S This string CTRL Z -5,+5L
```

You can insert a control character (e.g., **CTRL** C) into text by preceding it with **CTRL** V, then typing the letter, (e.g., C) without **CTRL**. **CTRL** V tells MS-DOS to

recognize the next letter as the corresponding control character.

You can also use a control character in any of the string arguments for the SEARCH or REPLACE commands using this method. For example:

S `CTRL` V Z is the command to search for the first occurrence of `CTRL` Z in a file.

R `CTRL` V Z `CTRL` Z foo is the command to replace all occurrences of `CTRL` Z with "foo"

S `CTRL` V C `CTRL` Z bar is the command to replace all occurrences of `CTRL` C with "bar"

To insert `CTRL` V into the text, simply type

```
CTRL V V
```

The `CTRL` Z character is read as an end-of-file mark by EDLIN. If `CTRL` Z appears in your file, you use the /B switch to tell EDLIN to ignore these control characters and show you the entire file.

The current line is the last line edited or the line currently being edited. EDLIN marks the current line with an asterisk (*). For example:

```
1: This is the first line of my new file.  
2: *
```

The MS-DOS Editing keys (explained in the chapter titled *Using the MS-DOS Keyboard*) function identically in EDLIN.

The underscore represents the cursor.

As with MS-DOS commands, EDLIN commands become effective only after you press `Enter`.

EDLIN Syntax The following syntax notation will be used in this chapter.

< > Parameters enclosed by angle brackets represent data you must enter. When the brackets enclose lower-case text, for example <line>, you must supply the entry defined by the text. When the angle brackets enclose upper-case text, for example <ENTER>, you must press the key named by the text.

[] Information enclosed in square brackets is optional.

CAPS Letters and words in capital letters are commands or portions of statements that must be entered exactly as shown.

EDLIN Command Options

EDLIN commands that use options are entered in the following form:

```
[<parameter>] <COMMAND> [<parameter>]
```

EDLIN parameters are:

- <line> Indicates you must specify a line number. Three possible entries can represent this parameter:
1. Enter a decimal integer from 1 to 65529. If you specify a number greater than the number of lines in the file, the line number is equal to the number after the last line number in the file.
 2. Enter a pound sign (#) to specify the line after the last line in memory. This has the same effect as specifying a number greater than the number of lines in memory.
 3. Enter a period (.) to specify the current line. The current line indicates the location of the last change to the file, but it is not necessarily the last line displayed. The current line is marked by an asterisk (*) between the line number and the first character of text in the line. For example:

```
10:*FIRST character of text
```
- <n> Indicates you must specify the number of lines. This parameter is used only with the WRITE and APPEND commands, which are used only if the file to be

edited is too large to fit in memory. Thus, when you use the <n> parameter, you enter the number of lines you wish to load from or write to disc.

<string> Indicates you must enter one or more characters to represent text to be searched for, replaced, or deleted. This parameter is used with the SEARCH and REPLACE commands. Each <string> must be terminated by **CTRL** Z or **Enter** (see the REPLACE command for details). No spaces should be left between a string and its command letter, unless you want those spaces to be part of the string.

Summary

In this section, you learned how to access EDLIN and create and edit text files; and how to enter EDLIN commands. The individual commands are described on the following pages.

EDLIN Commands

The following EDLIN commands are described in detail on the following pages:

Command	Purpose
<line>	Edits the specified line
A	Appends lines to your file
C	Copies lines from one location to another location in your file
D	Delete lines

E	Ends your editing session and writes your file to disc
I	Inserts lines into your file
L	Lists lines
M	Moves lines from one location to another
P	Lists your file one page (23 lines) at a time
Q	Quits editing without saving your file
R	Replaces lines in your file
S	Searches for specified strings
T	Transfers text from a file on disc to the current file
W	Writes a specified number of lines to disc

The Append Command

Purpose

Append adds the specified number of lines from a disc file to the end of the file being edited in memory.

Syntax

```
[<n>]A
```

n is the number of lines you wish to append from your disc file to the file in memory.

Operation

When you start EDLIN and specify an existing file, EDLIN will attempt to load it into memory; EDLIN loads lines until memory is 75% full to allow room to add lines during editing.

If your file is too large to load, you will have to edit all or a portion of the lines in memory, and save the edited lines to disc in order to free memory for more lines. Then, you can load unedited lines from disc into memory using the Append command. (Refer to the Write command for information on how to save edited lines to disc.)

If you do not specify the number of lines to append, lines are appended to memory until available memory is 75% full; no action will be taken if available memory is already 75% full.

When the Append command has read the last line of a file into memory, the message

```
End of input file
```

is displayed.

Example

To load 23 lines from the specified disc file to memory,
enter

23A

The Copy Command

Purpose

The Copy command copies a range of lines to a specified line number location in your file. If you use the <count> option, the Copy command can be repeated a specified number of times.

Syntax

```
[<fline>],[<lline>], <dline>,[<count>]C
```

fline is the first line in the range of lines to be copied

lline is the last line in the range of lines to be copied

dline is the line number location (destination) where lines are to be copied; data is placed before this line.

count is the number of times the specified range is to be copied; count must be an unsigned integer

Operation

If the <count> parameter is not specified, lines are copied once.

If either first line or last line is omitted, the default is the current line. This results in the current line being copied to that location.

Your file is renumbered automatically after the copy is completed, and the first of the copied lines becomes the current line. For example:

```
1,5,8C
```

copies lines 1 through 5 to line 8 and line 8 becomes the current line.

If line numbers overlap, you will receive the following error message:

```
Entry error
```

You must retype your copy command to recover. For example, the command

```
3,20,15C
```

is in error because it instructs EDLIN to copy lines 3 through 20 and insert them before line 15.

When copying is completed, EDLIN displays the * command prompt.

Examples

Assume the following file is in memory and ready to edit:

```
1: People usually point at what they want,  
2: which is why Hewlett-Packard developed  
3: HP Touch -- a special touch sensitive  
4: screen that's available with every  
5:*HP Vectra.
```

You can copy this entire block of text by entering

```
1,5,6C
```

The result is:

```
1: People usually point at what they want,
2: which is why Hewlett-Packard developed
3: HP Touch -- a special touch sensitive
4: screen that's available with every
5: HP Vectra.
6:*People usually point at what they want,
7: which is why Hewlett-Packard developed
8: HP Touch -- a special touch sensitive
9: screen that's available with every
10: HP Vectra
```

To copy lines and place them within this text, you must specify <dline> to tell EDLIN where to put them. If you want to copy lines 2 through 5 and place them before line 7, enter

```
2,5,7C
```



The resulting file will look like this:

```
1: People usually point at what they want.
2: which is why Hewlett-Packard developed
3: HP Touch -- a special touch sensitive
4: screen that's available with every
5: HP Vectra.
6: People usually point at what they want,
7:*which is why Hewlett-Packard developed
8: HP Touch -- a special touch sensitive
9: screen that's available with every
10: HP Vectra.
11: which is why Hewlett-Packard developed
12: HP Touch -- a special touch sensitive
13: screen that's available with every
14: HP Vectra
```

Notice that in both examples, lines have been renumbered by EDLIN.

The Delete Command

Purpose

The Delete Command deletes a specified range of lines in a file.

Syntax

```
[<fline>] [,<lline>] D
```

fline is the first line in the range of lines to be deleted

lline is the last line in the range of lines to be deleted

Operation

EDLIN supplies default values for the first line and the last line if either one or both are omitted. If you omit the first line, as in

```
 ,<line>D
```

deletion begins with the current line and ends with the line specified by <lline>. Note that the comma is required to indicate the missing first line parameter.

If you omit the last line parameter, as in

```
<line>D or <line> ,D
```

only the specified line is deleted.

If you omit both parameters, as in

`D`

only the current line is deleted, and the line following the deleted line becomes the current line. The line following the deleted range becomes the current line even if the deleted range includes the last line in memory. The current line, and any following lines, are renumbered.

When deletion is complete, EDLIN displays the * command prompt.

Examples

Assume that the following file exists and is ready to edit:

```
1: This sample has been written
2: to demonstrate dynamic line number
   generation
3: See what happens when
4: you use Delete and Insert
.
.
.
25: (the D and I commands)
26: to edit the text
27:*in your file.
```

To delete multiple lines, enter:

```
5,24D
```

The result is:

```
1: This sample file has been written
2: to demonstrate dynamic line number
   generation.
3: See what happens when
4: you use Delete and Insert
5:*(the D and I commands)
6: to edit the text
7: in your file.
```

To delete a single line, enter

```
LD
```

The result is:

```
1: This sample file has been written
2: to demonstrate dynamic line number generation
3: See what happens when
4: you use Delete and Insert
5: (the D and I commands)
6:*in your file.
```

To delete a range of lines from the following file:

```
1: This sample file has been written
2: to demonstrate dynamic line number generation
3:*See what happens when
4: you use Delete and Insert
5: (the D and I commands)
6: in your file.
```

enter the following EDLIN command:

```
LD
```

The result is:

```
1: This sample file has been written
2: to demonstrate dynamic line number generation
3:*in your file.
```

Notice that the lines are automatically renumbered.

The Edit Command

Purpose

The Edit command allows you to edit one line of text.

Syntax

```
[<line>]
```

line is the number of the line you want to edit; it may be expressed as a decimal integer, a pound sign (#), a period (.), or .

Operation

Any of the entries for <line>, described in the section on EDLIN Command Options, can be used depending on the line you wish to edit.

When you have specified a line number and pressed , EDLIN will display the line; below the line to be edited, EDLIN will display the line number. If you only press , EDLIN will assume you meant the line after the current line.

To replace the line with a new line, simply type the new line and enter it into your file by pressing . If you wish to edit the line without retyping it, you can use any of the MS-DOS editing keys described in the chapter titled *Using the MS-DOS Keyboard*. The existing line serves as the template until the key is pressed.

If you decide not to edit the line, you can keep the present line by positioning the cursor under the first character or past the last character on the line and pressing . Use the cursor control keys to move the cursor.

Caution

If you position the cursor by entering characters, including spaces entered by pressing the space bar, you will lose all or part of the data on the line. Be sure to use the cursor control keys to position the cursor.

Example

Assume the following file exists and is ready to edit:

```
1: This is a sample file created
2: to demonstrate what happens
3: when you edit line
4:*four.
```

To edit the fourth line, enter the line number, press , and see the following display:

```
*4
  4:*four.
  4:*
```

To edit the line using the MS-DOS Editing Keys, enter:

Function	Edit Keys	EDLIN response
Enter Insert mode	<input type="text" value="Ins"/>	
Type in the data	number__	4: number
Copy All	<input type="text" value="F3"/>	4: number four.
	<input type="text" value="Enter"/>	*

Your file will now read:

```
1: This is a sample file created
2: to demonstrate what happens
3: when you edit line
4:*number four.
```

You can either continue to add lines starting with line 5, or leave the edit mode by pressing .

The End Command

Purpose

The End command ends the editing session, saves the edited file on disc, and returns you to the MS-DOS command prompt.

Syntax

E

Operation

The End command saves the edited file on disc under the pathname entered when you accessed EDLIN, renames the original input file by adding the .BAK extension, and exits EDLIN. If the input file was created during the editing session, no .BAK file is created.

Because the End command has no options, you cannot tell EDLIN on which drive to save your file. Your file will be saved on the disc in the active drive unless you named a different drive in the path name.

Caution



You must be certain that your disc contains enough free space for your entire file. If there is not enough space available, the save operation will be aborted and all or part of your file will be lost. In this case, your original file will not be renamed with a filename extension of .BAK, and the portion of data that was saved to disc, if any, will have a filename extension of .\$\$\$.

When your End command has been executed, you will be returned to the MS-DOS command prompt.

The Insert Command

Purpose

The Insert command inserts lines of text immediately before the specified line. When you create a new file, the Insert command allows you to begin writing (inserting) lines.

Syntax

```
[<line>] I
```

line is the line number before which you wish to insert lines

Operation

When you enter the <line> Insert command, lines are inserted immediately before the line number specified. Entering the Insert command without specifying <line>, or by specifying <line> as a period (.), causes lines to be inserted immediately before the current line. Successive line numbers appear automatically each time a line is entered by pressing .

When the insert is completed, the line immediately following the inserted lines becomes the current line, and all line numbers following the inserted lines are incremented by the number of lines inserted.

If you specify a line number greater than the last line in your file, or if you specify the pound sign (#) as the line number, lines are inserted after the last line in memory. In this case, the last line inserted becomes the current line.

Press and simultaneously to leave Insert mode and return to the EDLIN prompt.

Examples

Assume the following file exists and is ready to edit using the Insert command:

```
1: This is a sample file used to demonstrate
2: dynamic line number generation.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7:*in your file.
```

To insert text before a specific line that is not the current line, enter

```
*4I
4:*_
```

and insert two new lines of text:

```
4:*FIRST NEW LINE OF TEXT
5:*SECOND NEW LINE OF TEXT
6*
```

and press `CTRL Break`. Now enter L to list the file. The result is:

```
1: This is a sample file used to demonstrate
2: dynamic line number generation
3: See what happens when you use
4: FIRST NEW LINE OF TEXT
5: SECOND NEW LINE OF TEXT
6:*Delete and Insert
7: (the D and I commands)
8: to edit text
9: in your file.
```

If the two lines that were inserted had been placed at the beginning of the file, the file would look like this:

```
1: FIRST NEW LINE OF TEXT
2: SECOND NEW LINE OF TEXT
3:*This is a sample file used to demonstrate
4: dynamic line number generation.
5: See what happens when you use
6: Delete and Insert
7: (the D and I commands)
8: to edit text
9: in your file.
```



If the two inserted lines had been placed at the end of the file (#I), the file would look like this:

```
1: This is a sample file used to demonstrate
2: dynamic line number generation.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.
8: FIRST NEW LINE OF TEXT
9:*SECOND NEW LINE OF TEXT
```

Note placement of the * to mark the current line in all of the above examples.

The List Command

Purpose

The List command displays a specified range of lines; the current line remains unchanged.

Syntax

```
[<fline>] [<,lline>] L
```

fline specifies the first line in the range to be listed

lline specifies the last line in the range to be listed

Operation

If you enter the List command with both options specified, the specified range will be listed. Default values are supplied by EDLIN if either one or both of the options are omitted.

If you omit both options, 23 lines will be displayed — 11 lines before the current line, the current line, and the 11 lines following the current line. If there are fewer than 11 lines before the current line, additional lines following the current line will be displayed to make a total of 23 lines.

If you omit the first line option and specify the last line option, as in:

```
,<line> L
```

listing will begin 11 lines before the current line and end with the specified last line. (Note that the beginning comma is required to indicate the omitted first option.) If the specified last line is more than 11 lines before the current line, the listing will be the same as if you had omitted both options.

If you omit the last line option, as in

```
<fline> L
```

23 lines will be displayed, starting with the specified <fline>.

Examples

Assume the following file exists and is ready to list:

```
1: This is a sample file used to demonstrate
2: dynamic line number generation.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
.
.
.
15:*The current line contains an asterisk.
.
.
.
26: to edit text
27: in your file.
```

To list a range of lines without reference to the current line, enter:

```
2-5L
```

The result is:

```
2: dynamic line number generation.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
```

To list a range of lines beginning with the current line, enter:

```
.-26 L
```

The result is:

```
15:*The current line contains an asterisk.  
.  
.(11 lines)  
.  
26: to edit text
```

To list a range of 23 lines centered around the current line (line 15 in our example), enter:

```
L
```

The result is:

```
4: Delete and Insert  
5: (the D and I commands)  
.  
.  
.  
13: The current line is listed in the middle.  
14: The current line remains unchanged.  
15:*The current line contains an asterisk.  
.  
.  
.  
26: to edit text.
```

The Move Command

Purpose

The Move command moves one line or a range of lines to the line specified.

Syntax

```
[<fline>],[<lline>],<dline> M
```

fline is the first line in the range of lines to be moved

lline is the last line in the range of lines to moved

dline is the line number location (destination) of the first line in the range of lines to be moved

Operation

If either the first line or last line parameter is omitted, it will default to the current line. When the move has been completed, the first line moved becomes the current line, and lines are renumbered according to the direction of the move. For example, the command

```
^+25^100M
```

moves text in the range current line/current+25 to line 100, and line 100 becomes the new current line.

If the line number parameters overlap, EDLIN will display the following error message:

```
Entry Error
```

To recover, retype the command using correct parameters.

Example

To move lines 20 through 30 to line 100, enter

```
20,30,100 M
```

The Page Command

Purpose

The Page command lists a specified range of lines one page (23 lines) at a time.

Syntax

```
[<fline>][,<lline>] P
```



fline is the first line in the range of lines to be listed

lline is the last line in the range of lines to be listed

Operation

The Page command pages through a range of lines or an entire file displaying 23 lines at a time. It differs from the List command in that it changes the current line to the last line displayed.

If the first line parameter is omitted, it defaults to the current line plus one. If the last line parameter is omitted, 23 lines are listed, and the new current line becomes the last line displayed.

Example

To display the first 23 lines of your file, and move the current line to the last line listed, enter:

```
1 P
```

The current line is the last line displayed.

The Quit Command

Purpose

The Quit command ends the current editing session without saving any changes or additions.

Syntax

`Q`

Operation

When you enter the Quit command, EDLIN prompts you to make certain you do not wish to save any changes made during the current editing session. The following message is displayed:

```
Abort edit (Y/N)?
```

Type Y if you want to quit the editing session. No editing changes will be saved, and no .BAK file will be created. (Refer to the End command in this chapter for more information about the .BAK file.)

Type N, or any alphanumeric character except Y, if you wish to continue the editing session.

Caution



When you start EDLIN and enter an existing filename, EDLIN erases the previous copy of the file with the filename extension of .BAK to make room to save the new copy. If you respond Y to the `Abort edit (Y/N)?` message, the backup copy of your file will be deleted.

The Replace Command

Purpose

The Replace command replaces all occurrences of a string of text in the specified range with a specified string of text, or with blanks.

Syntax

```
[<fline>][,<lline>][ ? ] R [<string1>]  
[ [CTRL] Z <string2>]
```

- | | |
|--|--|
| fline | specifies the first line to search for <string1> |
| lline | specifies the last line to search for <string1> |
| ? | instructs EDLIN to display an "OK?" prompt to verify correctness of each replacement |
| <string1> | is the text to be replaced |
| CTRL Z | separates <string2> from <string1> |
| <string2> | is the text to replace <string1> |

Operation

If either one or both of the line parameters is omitted, EDLIN supplies default values. If you omit the first line parameter, searching begins with the line after the current line. If you omit the last line parameter, the search ends with the last line in memory. If you omit both line parameters, EDLIN will search from the line following the current line to the last line in memory.

The Replace command displays changed lines each time they are changed. You can specify the optional ? parameter to request the “o.k.?” prompt after each display of a modified line. This allows you to type Y (yes) if you want to keep the line in its modified form. Enter any other alphanumeric character except Y if you do not want the the modification; the line will be kept in its original form. In either case, the search continues for further occurrences of the first string within the range of lines, including multiple occurrences within the same line.

EDLIN has precise requirements for the syntax of the Replace command. <String1> begins with the character position immediately following the R (Replace command letter), and continues until you press **CTRL** Z simultaneously, or until you press **Enter** if <string2> is omitted.

<string2> begins immediately after **CTRL** Z and continues until you enter the replacement string by typing **Enter**.

If <string1> is terminated with **CTRL** Z and <string2> is omitted, <string2> will be read as an “empty” string. For example:

```
R<string1> CTRL Z Enter
```

will delete all occurrences of <string1> in the specified range.

If you omit both <string1> and <string2>, EDLIN will re-use the search string (<string1>) entered with the most recent Search or Replace command, and will re-use the replacement text string (<string2>) entered in the last Replace command. Unless you are certain you know the contents of your previous Search and Replace commands, it is a good idea to supply these parameters.

When the Replace operation is complete, the last line changed becomes the current line.

You can place more than one command on a line containing a Replace command by terminating the replacement string with `CTRL Z`; then begin the next command in the following character position.

Example

Assume the following file is ready to edit using the Replace command:

```
1: This is a sample file
2: used to demonstrate dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.
8: The insert command can place new lines
9: in the file; there is no problem
10: because the line numbers are dynamic;
11: they will go all the way to 65529.
12:*Hi Mom, Hi Dad
```

To replace the string "and" with the string "or" in lines 2 through 12, enter:

```
2,12 Rand CTRL Z or Enter
```

The result, which is not very satisfactory, follows:

```
4: Delete or Insert
5: (the D or I commors)
8: The insert commor can place new lines
```

Because of the syntax requirements of this command, EDLIN does not differentiate between individual words and strings within words; for this reason, it is a good idea to use the `?` parameter to check and verify replacements before they are made. The same command entered with the `?` parameter

2.12? Rand CTRL Z or Enter

will be executed in the following way:

```
4: Delete or Insert
0.K.?Y
5: (The D or I commands)
0.K.?Y
5: (The D or I commors)
0.K.?N
8: The insert commor can place new lines
0.K.?N
*_
```

The result will be far more satisfactory. You can use the list command to check the results:

```
.
.
4: Delete or Insert
5: (The D or I commands)
.
.
.
8: The insert command can place new lines
```

The Search Command

Purpose

The Search command searches a range of lines to locate a specified string of text.

Syntax

```
[<fline>] [ , <lline> ] [ ? ] S <string>
```

fline	is the first line in the range of lines to be searched
lline	is the last line in the range of lines to be searched
?	instructs EDLIN to prompt "O.K.?" when the search string is found
string	is a string to be searched for and matched; it must be terminated with <input type="text" value="Enter"/>

Operation

The string you wish to have matched by the Search command must be exact as to upper- and/or lower-case letters. If you enter a search string in all upper-case, only upper-case occurrences will be matched; if you enter a search string in all lower-case, or in a combination of upper- and lower-case, only specific occurrences will be matched.

When the first line to contain the specified string has been found and displayed, the search will end unless you have used the ? optional parameter. The first line found that matches the specified string becomes the current line. If no match is found, the message `Not found` will be displayed.

If the ? option is included in the command, EDLIN will display the first line with a matching string; it will then prompt you with the message "O.K.". If you press either Y or , the line will become the current line and the search will terminate. If you press any other alphanumeric key (except Y), the search will continue until another match is found, or until all lines have been searched and the Not found message has been displayed.

If you omit either the first line or last line parameters, the system provides default values. If you omit the first line parameter, <fline> defaults to the line following the current line. If you omit the last line parameter, <lline> defaults to the last line in memory. If you omit both line parameters, the system searches from the line following the current line to the last line in memory.

If you do not enter a string, the Search command will use the last search string that was entered in a Replace or Search command. Unless you are certain you know the content of the previous Replace and Search commands, it is a good idea to supply these parameters. If the specified string is not found, the search ends and the Not found message is displayed. The current line remains unchanged.

EDLIN has exacting syntax requirements for the Search command. The search string must begin at the character position immediately following the S, and continue until you end the string by pressing .

You can place more than one command on a line containing a Search command by terminating the Search command with Z; then begin the next command in the following character position.

Examples

Assume that you want to edit the following file using the Search command:

```
1: This is a sample file that
2: will demonstrate
3: the Search Text command
4: using the
5: optional ? parameter
6: and the required <string>
7:*parameter.
```

To search for the first occurrence of and in the file, enter:

```
1,7 Sand
```

or

```
1, Sand
```

or

```
1Sand
```

You will see the following display:

```
3: the Search Text command.
*
```

The command is terminated, and line 3 has become the current line. As with the Replace command, EDLIN has not differentiated between the word "and" and the letters and in the word command. As a result, this may not be the "and" you were searching for. You can continue the search by simply entering the letter S. The search will continue using the previous search string; searching will begin with the line following the current line, which is the line last found. Now, in response to your search command, you will see:


```
*1,7 Sand
  3: the Search Text command
*S
  6: and required string
*
```

Another occurrence of "and" is found and the command is terminated; line 6 has now become the current line.

You can also search for strings using the ? parameter, which causes each matching line to be displayed until the specified range is exhausted. For example:

```
*1,7 ? Sand
  3: the search Text command.
0.K.? N
  6: and required string
0.K.? Y
*
```

The search command is terminated and line 6 becomes the current line.

The Transfer Command

Purpose

The Transfer command merges the contents of a specified file to a specified location in the file currently being edited.

Syntax

```
[<line>] T [<pathname>]
```



line specifies the line number location where **<pathname>** will be merged with the current file

pathname specifies the file to be merged with the current file

Operation

The contents of **<pathname>** will be inserted ahead of the specified line in the file being edited, and all line numbers will be adjusted beginning with the first line merged.

If **<line>** is omitted, it defaults to the current line.

The file being merged is read from the current directory of the specified drive or the active drive. If a path was specified when you started EDLIN, that path will be the current directory for that drive for the duration of the current EDLIN session; subsequent Transfer commands for that drive must be satisfied from the same directory.

The Write Command

Purpose

The Write command writes a specified number of lines from memory to disc.

Syntax

```
[<n>] w
```

n is the number of lines, starting with line 1, you wish to write to disc.

Operation

When you start EDLIN, lines are read into your system's memory until memory is 75% full. The remaining portion of memory is kept available to add data during editing.

If your file is too large to fit into memory, or if it becomes too large as a result of editing, you must save (write) some lines on to your disc to create more room in memory for editing, or for the remainder of your EDLIN file. The Write command allows you to save a specified number of lines from memory on to your disc so you can add more lines to memory. Use the Append command, described earlier in this chapter, to add more lines.

Lines are saved (written on your disc) beginning with line 1. Lines remaining in memory are renumbered, starting with 1, by EDLIN.

EDLIN Error Messages

When EDLIN finds an error, one of the following error messages is displayed. The cause and remedy is given for each message.

Message: Cannot edit .BAK file--rename file

Cause: You attempted to edit a file with a filename extension of .BAK. .BAK files cannot be edited; this extension is reserved for backup copies of EDLIN files.

Remedy: If you need to edit a .BAK file, either RENAME or COPY the .BAK file giving it a new extension.

Message: No room in directory for file

Cause: When you attempted to create a new EDLIN file, either the file directory was full, or you specified an illegal disc drive or an illegal filename.

Remedy: Check the filename and disc drive designation for your EDLIN command; if the command is no longer on the screen, and you have not entered a new EDLIN command, you can recover the faulty command using the **F3** (Copy all) key.

If the command line does not appear to have illegal entries, run the CHKDSK command for the specified disc. If the status report shows that the current directory is full, you can specify a path to a different directory in your EDLIN

command; create a new directory using the MKDIR command, and specify the path to that directory; or change discs.

Message: Entry Error

Cause: The last command typed contained a syntax error.

Remedy: Check for the correct syntax and retype the command.

Message: Line too long

Cause: When you entered a Replace command, your replacement string extended the line length beyond the 253-character limit. EDLIN aborted your Replace command.

Remedy: Shorten the replacement line or divide it into two lines, and try the Replace command again.

Message: Disk Full--file write not completed

Cause: You entered an End command and there was not enough space on the specified disc for the entire file. EDLIN aborted the E command, exited EDLIN, and returned you to MS-DOS COMMANDS. Use the DIR command to see if part of the file was written to disc; it will have an extension of .\$\$\$.

Remedy: Unfortunately, there is no way to recover the edited data. Even though there is a directory entry for your file, you will not be able to access the data when you run

EDLIN. You need to be certain that the default disc, or the disc specified in your EDLIN command, has sufficient free space for your file prior to beginning your editing session.

Message: Invalid drive name or file

Cause: You have specified an invalid pathname or drive when starting EDLIN.

Remedy: Re-enter your EDLIN command using correct pathname and drive designations.

Message: Filename must be specified

Cause: You did not include a file name when you started EDLIN.

Remedy: Re-enter the command; the syntax to access EDLIN is:

EDLIN <pathname>

Message: Invalid Parameter

Cause: You specified a switch other than /B, which is the only valid switch available when starting EDLIN.

Remedy: Specify /B to instruct EDLIN to ignore end-of-file marks and process your entire file.

Message: Insufficient memory

Cause: There is not enough memory to run EDLIN.

Remedy: You must free some memory by writing files to another disc or by deleting files before restarting EDLIN.

Message: File not found

Cause: The pathname given for a Transfer command was not found.

Remedy: Enter a valid pathname when issuing a Transfer command. It may be necessary to return to the MS-DOS command processor and view your disc directory (the DIR command).

Message: Must specify destination number

Cause: A destination line number was not specified for a Copy or Move command.

Remedy: Re-enter the command with a destination line number.

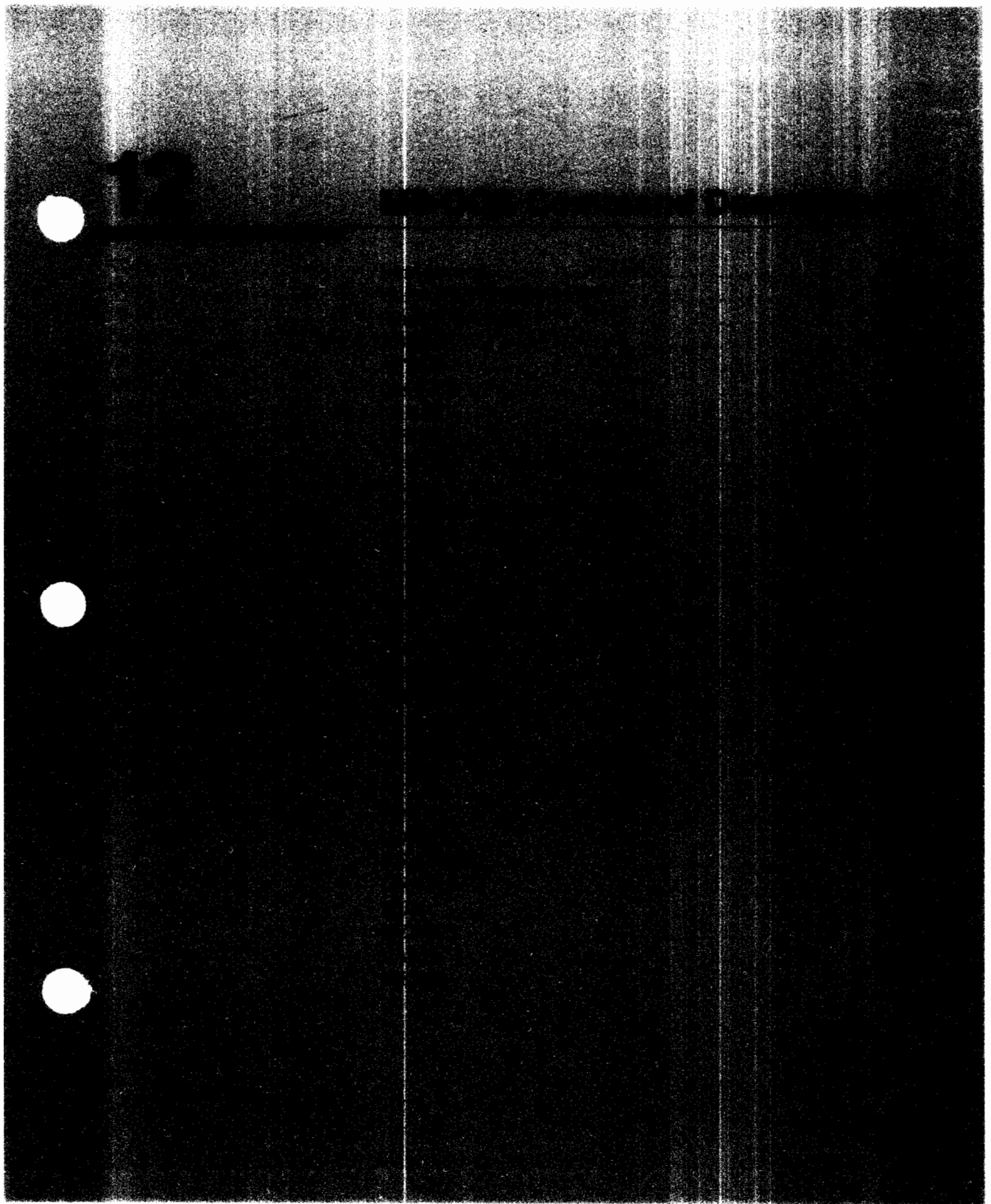
Message: Not enough room to merge the entire file

Cause: You attempted to execute a Transfer command when there was insufficient room in memory to hold the file.

Message: File creation error

Cause: The EDLIN temporary file cannot be created.

Remedy: Either your directory is full, or your file has the same name as a subdirectory in the directory the file is to be edited or located.



MODE	12-77
Video Display (Options 1 and 2)	12-77
Serial Communication (Option 3)	12-79
Parallel Printer Port (Options 4 and 5)	12-80
MORE	12-83
PATH	12-84
PRINT	12-86
PROMPT	12-90
RECOVER	12-93
RENAME	12-95
RESTORE	12-96
RMDIR	12-98
SET	12-99
SHARE	12-101
SORT	12-103
SUBST	12-105
SYS	12-108
TIME	12-110
TREE	12-112
TYPE	12-114
VER	12-116
VERIFY	12-117
VOL	12-119

12

MS-DOS Command Descriptions

Introduction

This chapter contains descriptions of all MS-DOS commands. It is intended to be used as a reference to syntax, usage, and options. We assume that you have already read the material in the previous chapters, or that you are familiar with MS-DOS.

Command Descriptions

The commands in this chapter are listed in alphabetical order. Although they are not grouped by function, all related commands are cross-referenced. Each MS-DOS command has its own listing. Each list contains the following information about the command:

Purpose

The purpose of the command is explained. This is a brief description of what the command will do, and what objects it operates on, i.e. files, directories, etc.

Type

This identifies the command as an internal or external MS-DOS command.

Syntax

The command's syntax is described here including all parameters, both required and optional.

Operation

Examples are provided that illustrate the more common uses of each command.

Note(s)

Any additional items of information necessary are included here.

Related Commands

Any other MS-DOS commands which are related are listed here.

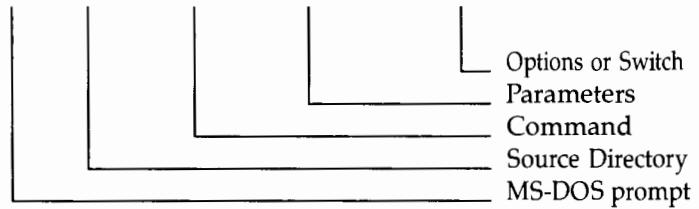
Syntax Notation

There are several special symbols and notations which we will use to describe the syntax of the various MS-DOS commands. The following is a list of these symbols and notations:

- Words in capital letters indicate commands or portions of commands that must be typed exactly as shown.
- Items enclosed in square brackets are optional. If you want to include the optional information, do not include the square brackets, only the information contained within the brackets.
- An ellipsis (...) means that you can repeat an item as many times as necessary.
- Items enclosed in angle brackets (< >) represent values which are supplied by the user.

- Items separated by a vertical bar (|) represent a list of possible choices. You must select one.
- You must include all punctuation exactly as shown (with the exception of the square and angle brackets).
- A command line consists of the directory, command, parameters, any options, and is terminated with `Enter`. A sample command line with its components is illustrated below.

```
C>A:\SUB1\CHKDSK A:\SUB2\FILE \F Enter
```



The MS-DOS command prompt usually contains the letter of the active drive followed by the "greater than" sign (>).

The Source Directory is optional. It consists of a drive designator and/or pathname, and is the directory where the command resides. Remember that a Source Directory is not valid in a command line for an internal MS-DOS command.

The Command is the name of the MS-DOS command itself. External commands are files with extensions of .COM or .EXE. The extension is not entered in the command line.



Note

You may type commands in any combination of upper- and lower-case letters; lower-case letters are converted by MS-DOS to upper-case prior to execution.

The Parameters are unique to each command. They typically consist of drive designators, pathnames, and filenames, as well as command punctuation.

Options or Switches are the final items in the command line. They consist of the forward slash character (/) followed by a single letter. Multiple options may be specified in a command line.

Several abbreviations will be used to shorten the command syntax notation. These abbreviations are listed below:

dir	Directory. This refers to the optional disc drive designator and/or pathname, and is equivalent to [<code><d></code> :[<code><path></code>].
sdir	Source Directory. This is the directory where the command resides.
subn	This refers to a subdirectory, where <i>n</i> is a number between 1 and 9. This will be used in commands where it is necessary to differentiate between two or more subdirectories. These subdirectories may or may not be on the same level.

leveln	This refers a subdirectory, where n is a number between 1 and 9 which represents the level of the subdirectory. This notation will be used with commands where the level of the subdirectory is important.
file name	These refer to valid filenames. In commands which require both a source and destination file, the source file will be designated <sfile>, and the destination file <dfile>.
sfile	
dfile	

The sample command line shown above would be notated in the following manner using these abbreviations.

```
[<sdir>]CHKDSK [<dir>][<file name>] [/F]
```

ASSIGN

Purpose

The ASSIGN command directs MS-DOS to substitute one drive designator for another during all subsequent disc or file references.

Type

ASSIGN is an EXTERNAL command.

Syntax

```
[<mdir>]ASSIGN [<x>[=]<y> [...]]
```

Operation

The ASSIGN command is used to instruct MS-DOS to change its drive designator to physical drive assignments. The *x* indicates the drive designator you want reassigned, and *y* is the drive designator of the physical drive you want all references to be made to. The command

```
ASSIGN A=C
```

will assign all operations specified as drive A: to drive C:. An application which uses flexible disc drives A: and B: only can be used on the hard disc drive C: by issuing the command

```
ASSIGN A=C B=C
```

Using ASSIGN without an instruction will terminate any drive designator reassignments in effect.

```
ASSIGN
```

Notes

1. Notice the examples above do not use the colon after the drive designator. If you type the colon, you will see this error message:
`Invalid parameter`
2. The ASSIGN program should only be used with application programs, and all reassignments should be terminated before using the MS-DOS BACKUP, LABEL, JOIN, PRINT, SUBST, and RESTORE commands.
3. The MS-DOS commands DISKCOPY and DISKCOMP ignore any drive reassignments.
4. The MS-DOS FORMAT command will issue an error message if an attempt is made to format an assigned drive.
5. Only physically resident drives may be assigned. For example, in a system with two flexible discs (A: and B:), a hard disc (C:), and a virtual disc using VDISK (D:), ASSIGN will issue an error message if an attempt is made to assign drive E:.

Related Commands

JOIN, PATH, SUBST

ATTRIB

Purpose

The ATTRIB command allows you to set or clear the READ-ONLY attribute flag for one or more files. ATTRIB may also be used to determine the setting of the READ-ONLY attribute flag of a specified file(s).

Type

ATTRIB is an EXTERNAL command.

Syntax

```
[<sdir>]ATTRIB [+R|-R] [<dir>]<file name>
```

Operation

The purpose of the READ-ONLY file attribute is to allow users to mark a file so that it can't be updated or erased. The file WORDPRO.COM can be protected with the command

```
ATTRIB +R WORDPRO.COM
```

To return the file to its normal state so it can be updated or erased, enter the command

```
ATTRIB -R WORDPRO.COM
```

The ATTRIB command without the +R or -R will return the state of the READ-ONLY flag for the file(s) specified. For example, the command line

```
ATTRIB *.COM
```

will result in the listing below:

```
R   C:\WORDPRO.COM  
    C:\ATTRIB.COM  
    C:\COMMAND.COM
```

Files with the letter "R" in the first column are READ-ONLY, and files without are not. Notice that the pathname of each file is displayed in the listing.

Notes

1. ATTRIB will accept wildcards in the file name. This allows you to set or clear the READ-ONLY flag on a group of files or all of the files in a directory.
2. When a file with the READ-ONLY attribute is copied using the COPY command, the new file does not have this attribute. Files copied with DISKCOPY preserve the state of the READ-ONLY attribute.

Related Commands

None

BACKUP

Purpose

The BACKUP command is used to make backup copies of the files stored on your Vectra's disc(s).

Type

BACKUP is an EXTERNAL command.

Syntax

```
[<sdir>]BACKUP<d>:[<path>] [<file name>]<d>:  
[ /S ][ /M ][ /A ][ /D:<mm-dd-yy> ]
```

Operation

The basic BACKUP command copies the files from the hard disc drive specified onto one or more flexible discs in the second drive specified. In its most basic form it will backup all files from the root directory of drive C: to a series of flexible discs in A:. The command line to do this is shown below:

```
BACKUP C: A:
```

This will backup all files in the current directory on the hard disc C: to a series of flexible discs in A:. You may specify a certain directory or even an individual file to backup, such as

```
BACKUP C:\WORDPRO A:
```

where WORDPRO could be either a subdirectory or an individual file.

Once the files have been copied to the backup discs, they may be copied back to the hard disc, or restored using the

RESTORE command. Refer to the description of the RESTORE command in this chapter for a detailed explanation.

BACKUP supports a set of options to provide additional flexibility to the backup procedure. These options are described below:

/S Backup files in all subdirectories. If this option is omitted, only the files in the specified directory (or current directory if the optional pathname is omitted) will be backedup. This option is commonly used since it is required if you wish to backup an entire disc that contains subdirectories. In order to backup the entire contents of the hard disc C: to a series of flexible discs in drive A:, the following command line would be executed.

```
BACKUP C: A: /S
```

/M Only files which have been modified since the last time BACKUP was run will be copied. This is referred to as incremental backup, and it will reduce the number of files which need to be copied.

/A Add the files being copied to the ones already on the backup disc. If this option is not specified, all existing files on the backup discs will be erased before any new ones are added.

/D Only files which have been modified on or after a specified date will be copied to the backup discs. This is another form of incremental backup.

- The date of the backup will be stored in the first line of the file.
- As each file is copied, the path, filename and backup disc number are entered into the backup log, one file per line. This information will be useful at a later date if you wish to restore a particular file (see the description of the RESTORE command).
- If a backup log file already exists, the current entries will be appended to the file.

Notes

1. All flexible discs must be formatted prior to using them with the BACKUP command. The discs should be formatted without the /S or /P options, since the system files will be erased by BACKUP (unless the /A option is specified with BACKUP).
2. Be certain that all drive reassignments (see ASSIGN command) are cancelled prior to starting a backup session.
3. BACKUP is not the same as COPY. The files on the backup discs are not in the standard MS-DOS format, and they can only be used by the RESTORE command.
4. Any files that are in a shared mode cannot be backed up (see SHARE command). For example, if the file CUSTOMER.DAT is in the shared mode, BACKUP will display the message

```
C:\CUSTOMER.DAT
Not able to backup at this time
```

Related Commands

RESTORE, COPY, DISKCOPY

BREAK

Purpose

Extends `CTRL-Break` and `CTRL-C` checking to include disc reads and writes. Normally, MS-DOS only checks for the `CTRL-Break` or `CTRL-C` during keyboard input, screen output, printer output, and auxiliary input or output. The BREAK command allows this to be extended to include disc operations as well.

Type

BREAK is an INTERNAL command.

Syntax

```
BREAK [ON|OFF]
```



Operation

To set the extended `CTRL-Break` and `CTRL-C` checking enter the command line

```
BREAK ON
```

To cancel extended checking enter

```
BREAK OFF
```

In order to determine whether extended checking is in operation, enter BREAK with no instruction. MS-DOS will display the current state of the extended checking.

Notes

1. The BREAK command may be included in the system configuration file (CONFIG.SYS). To change the default state to ON, include the command

`BREAK=ON`

anywhere in the file.

2. Some programs bypass MS-DOS for character and disc operations. When these programs are running, a `CTRL-Break` or `CTRL-C` might not interrupt the program, even though the BREAK ON command has been executed.

Related Commands

None

CHDIR

Purpose

The CHDIR command changes the current directory on the active or specified drive. The command may also be used to determine the path of the current directory.

Type

CHDIR is an INTERNAL command.

Syntax

```
CHDIR [<d>]<path>
```

Operation

The CHDIR command changes the current directory on a specified drive to a new path. If the optional drive designator is included, the current directory for that drive is changed to the new path; the current directory in the active drive is unchanged.

The following examples of CHDIR assume three subdirectories on the active drive. These subdirectories are described by the path

```
C:\SUB1\SUB2\SUB3
```

Assuming the current directory is SUB2, CHDIR can be used to change the current directory to SUB3 using the following command line

```
CHDIR SUB3
```

To return to the Root Directory enter

```
CHDIR \
```

To change the current directory to SUB2 enter

```
CHDIR SUB1\SUB2
```

Entering CHDIR with no parameters will display the path of the current directory. If the optional drive designator is included, the current directory for that drive will be displayed. The following command line displays the current directory on drive A:

```
CHDIR A:
```

Note

1. CD is an alternate form of the CHDIR keyword. The command line `CD \SUB1\SUB2` is equivalent to the command line `CHDIR \SUB1\SUB2`.

Related Commands

MKDIR, RMDIR, PATH

CHKDSK

Purpose

The CHKDSK command tests the integrity of a disc. It also reports information about the files on the disc. Additionally, it reports on system memory. If certain errors are encountered, CHKDSK may be used to correct them.

Type

CHKDSK is an EXTERNAL command.

Syntax

```
[<sdir>]CHKDSK [<dir>][<file name>] [/F][/V]
```

Operation

The CHKDSK command is used to determine the status of a disc or file(s). The following command line:

```
CHKDSK C:
```

will instruct CHKDSK to examine drive C:. CHKDSK cross-checks the root directory, any subdirectories, and the two copies of the File Allocation Table (FAT). If any errors are found, the appropriate error message is displayed. These messages, along with any recommended steps to correct the situation, are contained in the appendix titled *MS-DOS Message Directory*.

When CHKDSK has completed its inspection, it will display several statistics about the disc, its files, and the system memory. An example of this listing is shown below:

Volume HARDDISC

Created AUG 12, 1985 10:00

```
10592256 bytes total disk space
  40960 bytes in 2 hidden files
  45056 bytes in 11 directories
 4435968 bytes in 391 user files
  24576 bytes in bad sectors
6045696 bytes available on disk
```

```
655360 bytes total memory
220000 bytes free
```

This listing is representative of CHKDSK's display. The entries for volume, hidden files, directories (really subdirectories), user files, and bad sectors will not be listed if those items don't exist on the disc.

If you specify a filename instead of a disc drive, CHKDSK will determine how that file is stored on disc. It will report the number of non-contiguous blocks contained in the file. If the number of fragmented files on a disc (particularly a hard disc) grows too large, system performance can be degraded.

Wildcards may be used in the filename to determine the status of more than one file at a time. Note that CHKDSK only examines the files in one directory, either the current directory, or the one specified with the optional path. The following example is typical of this mode of CHKDSK:

```
C>CHKDSK \SUB1\*.TXT
```

```
Volume HARDDISC          Created AUG 12, 1983 10:00
```

```
10592256 bytes total disk space  
 40960 bytes in 2 hidden files  
 45056 bytes in 11 directories  
4435968 bytes in 391 user files  
 24576 bytes in bad sectors  
6045696 bytes available on disk
```

```
655360 bytes total memory  
220000 bytes free
```



```
C:\SUB1\CHAPT10.TXT  
  Contains 3 non-contiguous blocks.  
C:\SUB1\CHAPT11.TXT  
  Contains 2 non-contiguous blocks.  
C:\SUB1\CHAPT12.TXT  
  Contains 2 non-contiguous blocks.
```

CHKDSK supports two options: /F (fix) and /V (view). If CHKDSK finds any errors it will determine if it is able to fix them. If so, CHKDSK will ask you if you want them fixed. You may answer Yes or No, but you must have started CHKDSK with the /F option specified for CHKDSK to actually correct the errors.

If the /V option is specified, CHKDSK will display the names of all files and subdirectories as it verifies the integrity of the disc. This option is helpful for creating a list of all directories and the files contained in them.

Notes

1. CHKDSK does not accept a path without a filename as an instruction; it will attempt to interpret the path as a file. An example would be using CHKDSK to determine the amount of fragmentation in the files

contained in the subdirectory SUB1. The command line

```
CHKDSK C:\SUB1
```

will be incorrectly interpreted to mean “report on the file SUB1 in the root directory” and the error message

```
File not found
```

will be returned. The command line

```
CHKDSK C:\SUB1\*.*
```

will execute correctly.

2. You may use I/O redirection to direct CHKDSK's output to a device or file for later use. However, be sure not to use redirection and the /F option simultaneously.
3. If CHKDSK encounters lost clusters, it will ask you if you want them recovered. If you say yes, and the /F option has been used, CHKDSK will place each recovered allocation chain (one or more clusters) in a file with the name

```
FILEnnnn.CHK
```

where nnnn is a number starting with 0000. These files are placed in the root directory. You may look at these files to see if the information recovered is of any value, and erase them if not.

Related Commands

None

CLS

Purpose

The CLS command clears the display screen.

Type

CLS is an INTERNAL command.

Syntax

```
CLS
```

Operation

In order to clear the screen of all characters, enter the command line

```
CLS
```

Related Commands

None

COMMAND

Purpose

This command starts the MS-DOS command processor, COMMAND.COM.

Type

COMMAND is an EXTERNAL command.

Syntax

```
[<mdir>]COMMAND[<mdir>][/P][/C <command line>]
```

Operation

This command installs a copy of the command processor COMMAND.COM. This command may be typed from COMMAND.COM or PAM.

The optional source directory is searched if the file COMMAND.COM is not found in the directory(s) specified by the PATH variable in the environment string.

The /P option instructs MS-DOS to load this copy of COMMAND.COM permanently; i.e. the EXIT command will not return to the current command processor.

The /C option instructs the new command processor to execute the command line when it is loaded, then return to the original command processor. If the /C option is used, the /P option is ignored.

COMMAND is most frequently used from PAM. To enter the command processor COMMAND.COM permanently, enter the command line

```
COMMAND /P
```

at the PAM MS-DOS command prompt.

If the /P option had not been specified, PAM could be re-entered using the EXIT command at the MS-DOS command prompt.

Related Commands

EXIT, SHELL

COMP

Purpose

The COMP program compares the contents of two files and reports any differences.

Type

COMP is an EXTERNAL command.

Syntax

```
[<sdir>]COMP [<dir>][<file name1>]  
[<dir>][<file name2>]
```

Operation

The COMP program compares the files specified in the command line, and reports any differences. The command line

```
COMP C:\SUB1\FILE1.TXT B:\FILE2.TXT
```

will compare the contents of the file FILE1.TXT in the subdirectory SUB1 on drive C: with the file FILE2.TXT in the Root Directory of drive B:. If the filenames are not entered in the command line, COMP will prompt for them with the prompts shown below.

```
Enter primary file name
```

```
Enter 2nd file name or drive id
```

If the files are identical, the message

```
Files compare OK
```

will be displayed. If differences are discovered, they will be displayed in the following format:

```
Compare error at OFFSET xxxxx
File1 = yy
File2 = zz
```

where xxxxx is the byte offset from the start of the files, yy is the hex value of the byte in the first file, and zz is the hex value of the byte in the second file.

After 10 differences are found, COMP will stop comparing the files, display the error message

```
10 Mismatches - ending compare
```

and then display the prompt

```
Compare more files (Y/N)?
```



If you enter "Y" or "y", COMP will prompt for the names of the new files to compare with the following prompts:

```
Enter primary file name
```

```
Enter 2nd file name or drive id
```

Once the responses to these prompts have been entered, COMP will compare these files.

If you enter "N" or "n", you will return to MS-DOS.

Notes

1. If the two files do not contain the same number of bytes (as indicated in their respective directory entries) COMP will not compare them, but will display the error message

```
Files are different sizes
```

2. Wildcards may be used with COMP to compare more than two files. For example, the command line

```
COMP *.TXT *.BAK
```

would compare all files with an extension of .TXT with the respectively named file with the extension of .BAK. The names of the files are listed as they are compared.

3. If the second file name is omitted COMP will assume the filename of the first file.
4. If a file is not terminated with an End-of-file marker (**CTRL** Z), the message

```
EOF mark not found
```

will be displayed, but COMP will still compare the two files.

Related Commands

FC, DISKCOMP

COPY

Purpose

The COPY command is used to copy one or more files from one disc or directory to another. COPY can also be used to make a copy of a file in the same directory, but with a different name, or to concatenate two or more files into one.

Type

COPY is an INTERNAL command.

Syntax

The syntax for the COPY command has two different formats depending on the desired operation.

To copy a file:

```
COPY [<dir>]<sfile>[/A][/B]
      [<dir>][<dfile>][/A][/B][/V]
```

To concatenate two or more files into one:

```
COPY [<dir>]sfile1[/A][/B]
      [+ [<dir>]sfile2[/A][/B]...]
      [<dir>]dfile[/A][/B]
```

Operation

The COPY command copies the source file(s) to the desired destination. If the destination filename is not included in the command line, COPY assumes the destination file will have the same filename as the source file. In most cases, the destination will consist of a drive designator and/or pathname.

COPY supports three options; /V, /A, and /B. These are described below.

- /V Verify. This option instructs COPY to verify the file copied after it is written. This option performs the same function as the VERIFY command, except that it only operates during the COPY.

- /A Treat file as ASCII. When this option is specified for a source file, the file is copied up to the first end-of-file character (`CTRL Z`). The remainder of the file, if any, is not copied. When this option is used with a destination file, an end-of-file character is appended to the file.

- /B Treat file as Binary. When this option is specified for a source file, the entire file, based on the directory file size, is copied, including all end-of-file characters. When this option is used with a destination file, no end-of-file character is appended to the file.

When COPY is used to copy one file to another, the files are treated as Binary if no option is specified. COPY assumes files are ASCII when concatenating unless an option has been specified to the contrary.

In its most basic form, COPY copies a file from the current directory to the current directory of another drive. The following command line will copy the file INVOICE.DAT from the current directory on the active drive to the current directory on drive A:.

```
COPY INVOICE.DAT A:
```

Both the source and destination can be extended with drive designators and pathnames to operate on files in other directories than the current one. The following command line transfers INVOICE.DAT from subdirectory C:\SUB1 to A:\SUB2\SUB3.

```
COPY C:\SUB1\INVOICE.DAT A:\SUB2\SUB3
```

Wildcards may be used with COPY to move groups of files. The command line

```
COPY *.TXT A:
```

will copy all files with an extension of .TXT from the current directory on the active drive to the current directory on drive A:.

Files may be renamed during the copy process. If a filename is included in the destination which is different than the source filename, COPY will rename the file as it is copied.

```
COPY FILE1.TXT FILE2.TXT
```

This command line will copy the file FILE1.TXT to FILE2.TXT in the same directory.

COPY may be used to transfer data between system devices. A valid device name may be used as either the source or destination. The following example copies all characters from the console device to the file AUTOEXEC.BAT.

```
COPY CON AUTOEXEC.BAT
```

All characters entered at the keyboard will be stored until end-of-file (**CTRL** Z) and **Enter** are pressed. System devices may be used as source, destination, or both.

COPY may be used to concatenate two or more files into one. A good example of the usefulness of this form of the command would be a large manuscript containing many chapters. Usually, each chapter would be stored in its own separate file. If you need to combine them all, COPY will accomplish this with the following command

```
COPY CHPT1.TXT+CHPT2.TXT+CHPT3.TXT BOOK.TXT
```

The individual chapters (files) will be concatenated into the file BOOK.TXT. Any or all of the source or destination filenames may be preceded by an optional drive designator or pathname.

Notes

1. The attributes of the source file will not be transferred to the destination file.
2. Wildcards may be used with COPY to concatenate files. The example above could have been entered

```
COPY CHPT?.TXT BOOK.TXT
```

Caution

If wildcards are used, you must be certain that the destination filename is not included in the source filename reference. For example the command

```
COPY *.TXT BOOK.TXT
```

violates this restriction. An error will occur; however, it isn't detected until after the file BOOK.TXT is destroyed.

Related Commands

DISKCOPY, BACKUP

CTTY

Purpose

This command allows a character I/O device to be substituted for the standard console (keyboard and screen).

Type

CTTY is an INTERNAL command.

Syntax

```
CTTY <device name>
```

Operation

The CTTY command allows any valid character I/O device to be used as the MS-DOS Console. The device name may be any of the standard MS-DOS input/output devices, or an optional character device driver installed with the DEVICE command in the CONFIG.SYS file.

The command

```
CTTY AUX
```

will change all command I/O from the current device (the Console) to the AUX device (for example, another terminal). The command

```
CTTY CON
```

(entered from the AUX device) will change the command I/O back to the Console.

Notes

1. Command I/O redirection only applies to programs which use MS-DOS for Standard Input and Output. Many programs access the hardware directly, or use BIOS I/O routines; these programs will be unaffected by any assignments made by the CTTY command.
2. The assigned device must be a character I/O device, and must be capable of both input and output. Assigning command I/O to the PRN device (printer) for example, should not be done since the printer is incapable of sending characters to MS-DOS.

Related Commands

None



DATE

Purpose

The DATE command sets or displays the current system date.

Type

DATE is an INTERNAL command.

Syntax

```
DATE [<mm>-<dd>-<yy>]
```

Operation

The DATE command allows the MS-DOS system date to be set or displayed. To set the date, enter the command line

```
DATE MM-DD-YY or DATE MM/DD/YY
```

The system date is changed to the date specified. Note that either the "-" or "/" characters may be used as separators.

To display the current system date, enter the command line

```
DATE
```

MS-DOS displays the current date and asks for a new one with the following message:

```
Current date is Wed 05-15-85  
Enter new date [mm-dd-yy]:
```

If you press with no date, the date displayed will be retained as the system date.

Notes

1. The DATE command will check for invalid dates (i.e. Feb 30th, Feb 29th during non-leap years, etc.). The error message

`Invalid date`

will be displayed, and you will be prompted to re-enter a correct date.

2. The DATE display format may be modified by the COUNTRY command in the CONFIG.SYS file.
3. The DATE command sets the MS-DOS system date, but doesn't change the date kept by the Real-time clock. To change the Real-time clock's date, refer to the SETUP utility, or use PAM to change the date. The MS-DOS system date is set from the Real-time clock each time the system is reset.

Related Commands

TIME

DEL

Purpose

The DEL command is used to erase unwanted files.

Type

DEL is an INTERNAL command.

Syntax

```
DEL [<dir>]<file name>
```

Operation

The DEL command is used to erase one or more files from a disc. The command line consists of the DEL command followed by an optional drive designator and/or pathname, and the filename. If the filename doesn't contain wildcards, a single file is erased. The command line

```
DEL OLDFILE.DAT
```

will erase the single file OLDFILE.DAT in the current directory. More than one file may be erased by using wildcards. If the ** wildcard is used (erase all files) DEL will display the following message before executing the command line:

```
Are you sure (Y/N)?
```

If you answer "N", the command is cancelled and all files left intact. "Y" will erase all the files.

Notes

1. If a pathname is displayed without a filename, ****** is assumed and all files will be erased (after the **Are you sure prompt (Y/N)?** prompt). The following command line illustrates this:

```
DEL A:\
```

2. You may not delete a subdirectory with DEL; the RMDIR command must be used. However, a subdirectory must be empty before it can be removed, so the DEL command should be used first to erase the files in the subdirectory.
3. DEL is identical in function to the ERASE command.
4. The DEL command cannot be used to delete files with hidden or Read-Only attributes.

Related Commands

ERASE, RMDIR

DIR

Purpose

The DIR command lists the names of the files in a directory.

Type

DIR is an INTERNAL command.

Syntax

```
DIR [<dir>][<file name>][/P][/W]
```

Operation

The DIR command is used to determine what files are in a certain directory. If the optional directory and filename are omitted, the entire contents of the current directory will be displayed. After the files have been listed, the total number of files displayed and the amount of space remaining on the specified drive are listed.

```
Volume in drive C is HARD DISK
Directory of C:\

COMMAND  COM  23210  3-07-85  1:43p
FORMAT   COM  9398   3-07-85  1:43p
CHKDSK   COM  9435   3-07-85  1:43p
SYS      COM  3727   3-07-85  1:43p
DISKCOPY COM  4329   3-07-85  1:43p
          37 Files(s)  6348800 bytes free
```

A directory other than the current one may be specified in the command line. For example, if the current directory is the Root Directory on drive C:, the following command line will list the files in subdirectory SUB1 on drive A:

```
DIR A:\SUB1
```

A filename may also be specified in the command line. If the filename contains no wildcards, only the file with that name (if it is present in the directory) will be listed. Wildcards may be used in the filename to specify a group of files to be listed.

DIR supports two options: /P (page) and /W (wide). The /P option is useful with directories containing a large number of files. The /P option instructs DIR to pause at the end of every page. The listing will resume when any key is pressed.

The /W option displays the filenames in the wide format. The filenames are listed five to a line. When this option is selected, only the filename will be displayed; the time and date information is suppressed.

To display particularly large directories, the /P and /W options may be specified together, producing a listing in the wide format which pauses at the end of every page.

Subdirectories have special entries in the directory listing. The following example is the directory listing of a subdirectory. The . entry is the current subdirectory and the .. entry is the link to its parent directory. This subdirectory has another subdirectory nested in it. Subdirectories within the displayed directory are marked with a <DIR> in place of the file size.


```
Volume in drive D is VDISK V2.0
Directory of D:\LEVEL1

.                <DIR>  5-19-85  2:34p
..               <DIR>  5-19-85  2:34p
LEVEL2          <DIR>  5-19-85  2:34p
.
.
COMMAND  COM  23210  3-07-85  1:43p
FORMAT   COM  9398  3-07-85  1:43p
CHKDSK   COM  9435  3-07-85  1:43p
          41 File(s)  101888 bytes free
```

Notes

1. The amount of free space reported at the end of the listing is rounded down to the nearest 1024 bytes (1 KByte).
2. The display format of the time and date may be altered by the COUNTRY command in the CONFIG.SYS file. If a country other than the U.S. has been selected, the time and date format may be different than the examples shown.
3. DIR makes the following assumptions if a filename is specified without a name or without an extension.

DIR name = DIR name.*

DIR .ext = DIR *.ext

For example, to list all text files (extension of .TXT)

DIR.TXT

4. Hidden files (such as IO.SYS and MSDOS.SYS) will not be displayed in any directory listing, although they may reside in the directory.

Related Commands

None



DISKCOMP

Purpose

The DISKCOMP command compares the contents of two flexible discs sector for sector.

Type

DISKCOMP is an EXTERNAL command.

Syntax

```
[<sdir>]DISKCOMP [<d>: [<d>:]][/L]/B]
```

Operation

The DISKCOMP command will compare the contents of the flexible disc in the first drive with the disc in the second drive. The command line

```
DISKCOMP A: B:
```

will compare the flexible disc in Drive A: with the one in drive B: on a sector-by-sector basis.

When DISKCOMP starts, it will display the message

```
Insert FIRST diskette in drive A:
```

```
Insert SECOND diskette in drive B:
```

```
Strike any key when ready
```

If a compare error is discovered, the track and sector containing the mismatch will be displayed in the following format:

```
Compare error  
side xx, sector yy
```

where *xx* is the side number (0 or 1), and *yy* is the sector number.

After the compare operation is complete, DISKCOMP will ask if you want to compare more with the message

```
Compare more diskettes (Y/N)?
```

If you enter "Y" or "y" you will be prompted to insert the new discs. Entering "N" or "n" will return you to the MS-DOS prompt.

DISKCOMP supports two options.

- /1 If this option is specified, only the first side of the flexible discs will be compared, whether they are double-sided discs or not.
- /8 If this option is specified, only 8 sectors per track will be compared, whether the discs have 8, 9, or 15 sectors per track.

Notes

1. DISKCOMP may not be used on hard or virtual discs.
2. If both drive designators are omitted in the command line, a single drive compare on the active drive is assumed.
3. If the second drive designator is omitted from the command line, the active drive is assumed for the second drive.

4. Two discs will usually be identical only if one has been made from the other using DISKCOPY. If the COPY command has been used to transfer files from one disc to another, the discs will probably not compare, since COPY does not place data in the same place on the target drive as it was on the source drive. To compare discs made with COPY, refer to the FC and COMP commands.
5. DISKCOMP cannot be used on ASSIGNED drives.

Related Commands

COMP, DISKCOPY, FC

DISKCOPY

Purpose

DISKCOPY copies the contents of one flexible disc to another flexible disc.

Type

DISKCOPY is an EXTERNAL command.

Syntax

```
<mdir>DISKCOPY [<d>: [<d>:]] [/1]
```

Operation

The DISKCOPY command makes an exact copy of one flexible disc to another. The command line

```
DISKCOPY A: B:
```

will make an exact copy of the disc in drive A: on the disc in drive B:.

DISKCOPY supports one option.

/1 Only the first side of the disc is copied, whether the disc is single or double sided.

When DISKCOPY starts, it will display the message

```
Insert SOURCE disk in drive A:
```

```
Insert TARGET disk in drive B:
```

```
Press any key when ready . . .
```

When the copy is complete, DISKCOPY will display the message

Copy another (Y/N)?

If you enter "Y" or "y", you will be instructed to insert the next set of discs. Entering "N" or "n" will return you to MS-DOS.

Notes

- 1.** DISKCOPY may not be used on hard or virtual discs.
- 2.** If both drive designators are omitted in the command line, a single drive copy on the active drive is assumed.
- 3.** If the second drive designator is omitted from the command line, the active drive is assumed for the second drive.
- 4.** DISKCOPY cannot be used on ASSIGNED drives.
- 5.** If the target disc has never been formatted, DISKCOPY will format the target drive during the copy process.
- 6.** If disk errors are encountered on either the source or target disc, DISKCOPY will report the track, sector, and size of the error, and proceed with the copy process. The target disc may or may not be usable in this situation. DISKCOMP or COMP may be used to verify the state of the target disc.

Related Commands

COPY,DISKCOMP

ERASE

Purpose

The ERASE command is used to delete unwanted files.

Type

ERASE is an INTERNAL command.

Syntax

```
ERASE [<dir>]<file name>
```

Operation

The ERASE command is used to delete one or more files from a disc. The command line consists of the ERASE command followed by an optional drive designator and/or pathname, and the filename. If the filename doesn't contain wildcards, a single file is deleted. The command line

```
ERASE OLDFILE.DAT
```

will delete the single file OLDFILE.DAT in the current directory. More than one file may be deleted by using wildcards. If the ** wildcard is used (delete all files) ERASE will display the following message before executing the command line:

```
Are you sure (Y/N)?
```

If you answer "N", the command is cancelled and all files left intact. "Y" will delete all the files.

Notes

1. If a pathname is displayed without a filename, ** is assumed and all files will be deleted (after the Are

you sure prompt (Y/N)? prompt). The following command line illustrates this.

```
ERASE A:\
```

- 2.** You may not delete a subdirectory with ERASE; the RMDIR command must be used. However, a subdirectory must be empty before it can be removed, so the ERASE command should be used first to delete the files in the subdirectory.
- 3.** ERASE is identical in function to the DEL command.
- 4.** The ERASE command cannot be used to delete files with the hidden attribute.

Related Commands

DEL, RMDIR

EXE2BIN

Purpose

The EXE2BIN command converts files from .EXE format to .COM or .BIN format.

Type

EXE2BIN is an EXTERNAL command.

Syntax

```
[<sdir>]EXE2BIN [<dir1>] <sfile> [<dir2>]  
[<dfile>]
```

Operation

The EXE2BIN command may be used to convert program files from .EXE format to .COM or .BIN format. The command line specifies the file to be converted, and an optional filename for the output file. If the output name isn't specified, EXE2BIN will use the input filename with an extension of .BIN. To convert the file MYPROG.EXE from .EXE format, and create the file MYPROG.COM, enter the following command line:

```
EXE2BIN MYPROG MYPROG.COM
```

Note that the extension on the input file doesn't need to be specified; .EXE is presumed.

Notes

1. The input file must be in valid .EXE format as produced by the linker. The actual code and data portions of the program must be less than 64K combined, and there must be no STACK segment. If the input file is not in valid .EXE format, EXE2BIN

will issue the error message

File can't be converted

2. EXE2BIN produces two different types of output files depending on whether or not the initial CS:IP is specified in the .EXE file. These two cases are described below:

- If no CS:IP has been specified (i.e. the .EXE file contains a CS:IP value of 0:0), a pure binary conversion is assumed. This is the format which must be used to produce installable device drivers.

If segment fixups are necessary (that is, the program contains instructions requiring segment relocation), you will be prompted for the values. The resulting program will only be useable when loaded at the absolute memory address which you specify. The MS-DOS command processor will be unable to load the program.

- If the CS:IP is specified as 0000:100H in the .EXE file, it is assumed that the file will run as a .COM file, with the location pointer set at 100H by the assembler ORG statement. No segment fixups are allowed, because .COM programs must be segment relocatable. The requirements of a .COM program are explained in the *HP Vectra MS-DOS Programmer's Reference Manual*. The output file may be renamed with an extension of .COM (if this wasn't specified in the command line). The MS-DOS command processor can now load and execute the program in the same way as the .COM programs supplied on your MS-DOS disc.

Related Commands

LINK

EXIT

Purpose

The EXIT command allows you to exit the command processor and return to the previous command processor (if one exists).

Type

EXIT is an INTERNAL command.

Syntax

EXIT



Operation

The MS-DOS command processor COMMAND.COM may be loaded by PAM (or any other alternate command processor). In order to exit back to PAM, or the previous command processor, enter the command line

EXIT

Note

1. EXIT will perform no operation if no previous command processor is active.

Related Commands

COMMAND

FC

Purpose

The FC command compares the contents of two files.

Type

FC is an EXTERNAL command

Syntax

```
[<sdir>]FC [/A][/B][/C][/L][/LB <n>]
           [/W][/T][/N][/<nnnn>]
           [<dir1>]<file name1>
           [<dir2>]<file name2>
```

Operation

FC enables you to compare the contents of two files. Two types of comparisons can be made:

- source comparison (line-by-line)
- binary comparison (byte-by-byte)

FC matches the first file against the second and reports any differences between them. Both filenames can have pathnames attached to them. For example, the command line

```
FC B:\MEMO\FILE1.TXT C:\FILE2.TXT
```

takes FILE1.TXT in the \MEMO directory of disc drive B: and compares it with FILE2.TXT in the Root Directory of C:.

FC supports several options which control how the comparison is made. These options are listed below:

- /A abbreviates the output of a source (ASCII) comparison. Instead of displaying all of the lines that are different, only the beginning and ending lines of each set of differences are displayed. The intermediate lines are represented by ellipses (...).
- /B forces a binary comparison of both files, byte-to-byte. It is the default when you compare files with extensions of .EXE, .COM, .SYS, .OBJ, .LIB, or .BIN.
- /C instructs FC to ignore the case of letters and to consider all letters as upper-case. Use this switch for source comparisons only.
- /L compares the files in ASCII mode. It is the default when you compare files that do not have the extensions of .EXE, .COM, .SYS, .OBJ, .LIB, or .BIN.
- /LB <n> sets the internal buffer to <n> lines. The default length of the internal buffer is 100 lines.

Notes

If the number of differing lines is equal to or greater than *n*, FC aborts. The results of the file comparison, therefore, become invalid. In other words, *n* has to exceed the number of differing lines by 2.

- `/W` compresses white spaces created by tabs and spaces. Use this switch for source comparisons only.
- `/T` does not expand tabs to spaces. The default is to treat tabs as spaces to 8 column positions.
- `/N` displays line numbers on an ASCII comparison.
- `/<nnnn>` specifies the number of lines that must match after a difference is found for the files to be considered as matching again. In other words, FC will continue to list lines that are different between the two files until it encounters `<nnnn>` number of lines that are the same. The default is 2 and you can specify a number that is up to four digits in length.

Notes

1. Switches placed after the filenames will be ignored and the default values will be used. Incorrect or unrecognized parameters are ignored and do not prevent the FC command from executing.
2. For ASCII comparisons, FC displays the differing lines unless there are more consecutive, differing lines than there is room for in the internal buffer. You can use the `/LB` switch to regulate the line length of the buffer (default = 100 lines). For binary comparisons, the FC command displays all differing bytes.

Here are two sample text files for comparison:

ALPHA.TXT	BETA.TXT
all	all
good	good
things	things
must	come
eventually	to
come	some
to	kind
some	of
kind	end
of	
an	
end	

To compare the two sample files and display the differences on the screen, enter the command line

```
FC B:ALPHA.TXT B:BETA.TXT
```


FC now begins each display of differences with

- the file name
- the last matching line preceding a difference
- the different lines
- the next matching line

The source comparison of the two sample files shows the following:

```
*****B:ALPHA.TXT
things
must
eventually
come
*****B:BETA.TXT
things
come
*****
```

```
*****B:ALPHA.TXT
of
an
end
*****B:BETA.TXT
of
end
*****
```

To force a binary comparison of the two files, enter the command line

```
FC/B B:ALPHA.TXT B:BETA.TXT
```

The results are displayed in three columns:

First column	the relative address of the pair of bytes from the beginning of the file. Addresses start at 00000000.
Second column	the mismatched bytes from <filename1>
Third column	the mismatched bytes from <filename2>

A portion of a binary comparison of our two sample files is shown below. Note that if one file contains more data than the other, a message is displayed at the end of the comparison.

```
00000013: 6d 63
00000014: 75 6f
00000015: 73 6d
:      :
:      :
```



```
fc: B:ALPHA.TXT longer than B:BETA.TXT
```

To demonstrate the use of the line buffer switch (/LB <n>), the following example reduces the size of the line buffer to three lines. The sample files have more consecutive, differing lines than the buffer can hold and the comparison aborts. The default length of the line buffer is 100 lines.

To set the line buffer to three lines and compare the files, enter the command line

```
FC/LB 3 B:ALPHA.TXT B:BETA.TXT
```

The following message will result:

```
Resync failed. Files are too different.
```

The line number switch (<nnnn>) is set to six lines in this example. FC will not encounter six consecutive, matching lines after the first difference is found. It will continue to list lines until the end of one of the files is reached.

The command line

```
FC/L B:ALPHA.TXT B:BETA.TXT
```

will produce the comparison shown below.

```
*****B:ALPHA.TXT
things
must
eventually
come
to
some
kind
of
an
end
*****B:BETA.TXT
things
come
to
some
kind
of
end
*****
```

Related Commands

COMP

FIND

Purpose

FIND searches for a specified text string in one or more input files.

Type

FIND is an EXTERNAL command.

Syntax

```
[sdir]FIND [/V][/C][/N] <"text string">
          [[<dir>][<file name>...]
```

Operation

FIND is one of three filters provided with the MS-DOS operating system. FIND will look through one or more specified files and attempt to locate the search string.

The text string must be enclosed in quotes, and may contain any valid character code. Two double quotes will be interpreted as an embedded double quote, and not the end of the string (i.e. "He said, ""Hello"" to me." will look for He said, "Hello" to me.). FIND differentiates between upper- and lower-case letters, so the string "Hello" in a file will not match the search string "HELLO".

The following example displays all lines from the files BOOK1.TXT and BOOK2.TXT that contain the string "Fool's Paradise."

```
FIND "Fool's Paradise" BOOK1.TXT BOOK2.TXT
```

FIND supports three options which allow additional flexibility in searches. These options are described below:

- /V Instructs FIND to display all lines not containing the search string.
- /C Instructs FIND to count, instead of display, the number of lines containing the search string.
- /N Instructs FIND to precede each line displayed with its line number.

The /C option takes precedence over the /N option. If /N is specified with /C, /N will be ignored by FIND. Specifying both the /C and /V options will count the number of lines not containing the string.

If no file is specified, the Standard Input is used as the source. If no redirection is active, the input from the console is used (until the end-of-file character, `CTRL Z`, is input). The following example will display all the directory entries with a .TXT extension.

```
DIR > FIND 'TXT'
```

Related Commands

None

FORMAT

Purpose

The FORMAT command prepares a disc for use.

Type

FORMAT is an EXTERNAL command.

Syntax

```
[<sdir>]FORMAT [<d>:][/S][/P]
                    [/V][/B][/L][/4][/B]
```

Operation

The basic FORMAT command line consists of the keyword FORMAT and the drive to format. The following command line will format the disc in drive A:.

```
FORMAT A:
```

If no disc drive designator is contained in the command line, FORMAT will assume the active drive.

Caution



The FORMAT command will destroy any data contained on the disc being formatted. Be sure that the disc contains no valuable data before it is formatted.

FORMAT performs several tasks during the format operation. It formats the disc, checks for defective tracks, and constructs the root directory and File Allocation Tables. If a defective track is found, FORMAT will mark it as such, which will prevent MS-DOS from placing data in that track.

FORMAT supports several options, which are listed below.

- /S Place System on disc. If this option is selected FORMAT will place the two hidden MS-DOS system files, MSDOS.SYS and IO.SYS, on the disc as well as the command processor, COMMAND.COM.
- /P Place System and PAM on disc. If this option is selected FORMAT will place the MS-DOS system files listed above, along with the PAM files and a CONFIG.SYS file to start PAM, on the disc.
- /V Write Volume Label. FORMAT will prompt you for a volume label after the disc is formatted. A volume label may be up to 11 characters in length. The volume label is displayed by DIR, and may be read by a program via an MS-DOS system call.
- /8 Format 8 sectors per track. This option directs FORMAT to make 9 sectors on each track, and instructs MS-DOS to use only 8. This option is not valid when formatting hard discs or high capacity flexible discs.
- /1 Format Single Sided. This option instructs FORMAT to format the disc as a single-sided disc, regardless of the drive type. This option is not valid when formatting hard discs or high capacity flexible discs.
- /4 Format a 360Kb disc in a 1.2 Mb drive. This option formats a double-sided, standard capacity (360K) disc in a high capacity disc drive. This option is not valid when formatting hard discs or high capacity flexible discs.

/B Format for backwards compatibility. This option formats a disc with 8 sectors per track, and reserved space for the MS-DOS files IO.SYS and MSDOS.SYS. These file are not actually transferred; they must be transferred with SYS. The purpose of this option is to allow a disc to be formatted, and an earlier version of MS-DOS installed on it. This option is not valid when formatting hard discs or high capacity flexible discs.

Notes

- 1.** A fixed disc must be reformatted if the size of the MS-DOS partition is changed with FDISK.
- 2.** FORMAT will automatically determine a flexible drive's characteristics and will attempt to format accordingly (unless the /1, /4, or /8 option has been specified).
- 3.** If the /S option has been specified, the system files (MSDOS.SYS, IO.SYS, and COMMAND.COM) are transferred from the root directory of the active drive. If the active drive is a hard disc, and the files aren't present, FORMAT will prompt you to insert a system disc in drive A:.
- 4.** The parameters /8 and /V can't be specified with the /B option.

Related Commands

SYS, FDISK

GRAFTABL

Purpose

The GRAFTABL command loads the character fonts for additional characters for display by the Multi-mode Video display adapter in the graphics mode.

Type

GRAFTABL is an EXTERNAL command.

Syntax

```
[<mdir>]GRAFTABL
```

Operation

The GRAFTABL command loads an optional character set which can be displayed in the graphics mode. GRAFTABL loads a character font set for character codes 128 through 255. This command is useful for displaying non-U.S., technical, or other character fonts.

A portion of the GRAFTABL command remains resident in memory. When the command is run, the message

```
GRAPHICS CHARACTERS LOADED
```

is displayed, indicating that the resident portion has been loaded. GRAFTABL should only be loaded once. If an attempt is made to load GRAFTABL a second time, the attempt will fail.

Related Commands

GRAPHICS

GRAPHICS

Purpose

The GRAPHICS command allows the contents of a graphics screen to be output to a printer.

Type

GRAPHICS is an EXTERNAL command.

Syntax

```
[<mdir>]GRAPHICS [<printer>][[/R]][/B]
```

Operation

The GRAPHICS command loads support for outputting the contents of a graphics screen to the printer. After GRAPHICS has been run, the contents of a graphics screen may be printed by pressing **Shift** **Prt Sc**.

The GRAPHICS command supports 5 types of printers.

COLOR1	Color Printer with black ribbon
COLOR4	Color Printer with RGB (Red, Green, Blue, Black) ribbon
COLOR8	Color Printer with CMY (Cyan, Magenta, Yellow, Black) ribbon
COMPACT	Compact Printer
GRAPHICS	Graphics Printer

Support for any one of these printers may be selected by specifying the printer type in the command line. If the printer type is omitted, the default is the GRAPHICS printer.

The /R option reverses the black and white from the display, i.e. black is printed as white, and white printed as black.

The /B option will instruct GRAPHICS to print the background color. If the this option is omitted, the default is not to print the background.

Once the GRAPHICS command has been executed, screen images in the graphics mode may be output to the printer by pressing Shift Prt Sc.

Notes

- 1.** GRAPHICS increases the resident size of MS-DOS.
- 2.** Screen images displayed in the 640 X 200 mode are displayed sideways on the printer. All other graphics modes are printed normally.

Related Commands

GRAFTABL

JOIN

Purpose

The JOIN command allows a disc drive to be logically joined to a subdirectory in another drive.

Type

JOIN is an EXTERNAL command.



Syntax

```
[<sdir>]JOIN <d>: <dir>
```

or

```
[<sdir>]JOIN <d>:/D
```

or

```
[<sdir>]JOIN
```

Operation

The JOIN command has three formats. The first allows a disc to be joined to another, the second to delete a join, and the third to list the drives which are currently joined.

The disc in drive A: is joined to drive C: with the command line

```
JOIN A: C:\DRIVE_A
```

DRIVE_A is a new subdirectory on drive C:, so all of the data on the disc in drive A: is now in the directory structure of drive C:. All references to drive A: will produce the error message:

```
Invalid drive specification
```

A drive may only be joined at the root of its "host" drive. In the example above, drive A: was joined as a subdirectory in the root directory of drive C:. Drive A: could not be joined as a subdirectory of SUB1 in the example above, since this would be joining below the root level.

The subdirectory which the drive is to be joined to must be empty. If the subdirectory specified in the command line does not exist, JOIN will create it prior to joining the target drive.

A drive may be unjoined from another drive with the second form of the JOIN command. The following command line will unjoin drive A: from the assignment made above (drive C:).

```
JOIN A:/D
```

The third form of the JOIN command will display all drives currently joined. The joining of drive A: to drive C: described above would be displayed

```
A: => C:\DRIVE__A
```

If none of the drives in the system are joined, nothing will be displayed.

Notes

1. A join is only in effect until it is undone with a subsequent JOIN command, or until the system is reset. It does not permanently alter the directories of either drive.
2. It is recommended that you use the same subdirectory each time you join a disc since the subdirectory must be empty at the time the drives are joined, and the subdirectory is not automatically deleted when the discs are unjoined. This will prevent the root directory from becoming cluttered with empty subdirectories which have been used for joining.

3. JOIN cannot be used on a drive which has been SUBSTed, ASSIGNed, or redirected over the network.

Related Commands

ASSIGN, SUBST

KEYBUS

Purpose

The KEYBUS command replaces the ROM BIOS keyboard routines to provide support for non-U.S. keyboards.

Type

KEYBUS is an EXTERNAL command.

Syntax

```
[<mdir>] KEYBUS
```

Operation

The KEYBUS command loads support for five non-U.S. keyboards. KEYBUS adds resident code to MS-DOS to support the different layouts.

Once the command is executed, all keystrokes will be interpreted by the resident code. To alternate between the non-U.S. keyboards and the U.S. keyboard, the **CTRL**, **Alt** and Function keys (**F1** - **F8**) are used. To change from one language keyboard to another press the appropriate **CTRL**-**Alt**-Fn sequence, where Fn represents one of the function keys **F1** through **F8**.

The following table lists the five key sequences and their respective languages or countries.

Key Sequence	Country
CTRL Alt F1	United States
CTRL Alt F2	United States
CTRL Alt F3	United States

CTRL **Alt** **F4** United Kingdom

CTRL **Alt** **F5** Germany

CTRL **Alt** **F6** France

CTRL **Alt** **F7** Italy

CTRL **Alt** **F8** Spain

Pressing **CTRL** **Alt** **F8** will instruct the resident portion of KEYBUS to interpret the keyboard as Spanish. Pressing **CTRL** **Alt** **F1** will return to the U.S. keyboard characters.

Note

- 1.** The code is loaded above MS-DOS and remains resident in memory until the system is reset. KEYBUS increases the resident size of MS-DOS by approximately 3K.

Related Commands

COUNTRY

LABEL

Purpose

The LABEL command adds, modifies, or deletes the volume label on a disc.

Type

LABEL is an EXTERNAL command.

Syntax

```
[<sdir>]LABEL [<d>:][<volume label>]
```

Operation

The LABEL command provides a means to add or modify the volume label on a disc. The label is placed on a disc when it is formatted using the /V option (see FORMAT command). If the volume label wasn't specified, LABEL can add one. LABEL can also modify or delete existing volume labels. A volume label may be up to eleven characters in length.

If you don't specify the new volume label in the command line, LABEL will prompt you for one with the following message

```
Volume label (11 characters, ENTER for none)?
```

You may enter a new volume label, or simply press to delete the current volume label. If you enter more than 11 characters, only the first 11 are used. If the drive is not specified the active drive is assumed.

Note

1. LABEL should not be used on drives which have been SUBSTed, ASSIGNed, or redirected over the network.

Related Commands

FORMAT, VOL

MKDIR

Purpose

The MKDIR command is used to create new subdirectories on a disc.

Type

MKDIR is an INTERNAL command.

Syntax

```
MKDIR [<d>:]<path>
```



Operation

The MKDIR command creates the subdirectory specified by the pathname in the command line. The subdirectory is created on the specified drive. If no drive is specified, the active drive is assumed.

If the path starts with the “\” character, it is assumed that the path starts with the root directory, regardless of what directory is the current directory. Otherwise, the path is assumed to start with the current directory (which may or may not be the root directory).

The following example creates the subdirectory LEVEL1 in the root directory of drive C:.

```
C>MKDIR \LEVEL1
```

The following two examples create the subdirectory LEVEL2 in the subdirectory LEVEL1. The first example defines the path from the root directory, while the second assumes LEVEL1 is the active directory.

```
C>MKDIR \LEVEL1\LEVEL2
```

```
C>MKDIR LEVEL2
```

Notes

1. Subdirectories may be nested any level deep. However, any path from the root directory is limited to 63 characters.
2. MD is an alternate form of the MKDIR keyword. The command line MD \LEVEL1\LEVEL2 is equivalent to MKDIR \LEVEL1\LEVEL2.

Related Commands

CHDIR, RMDIR, PATH

MODE

Purpose

The MODE command is used to control the parameters for the Multi-Mode video and Serial/Parallel adapter cards.

Type

MODE is an EXTERNAL command.

Syntax

```
[<sdir>]MODE <display>
```

```
[<sdir>]MODE [<display>],<adjust>[ ,T]
```

```
[<sdir>]MODE COMn [ : ] <baud>  
[ ,<parity>[ ,<databit>[ ,<stopbit>[ ,P]]]]
```

```
[<sdir>]MODE LPTn [ : ] [<horz>]  
[ , [<vert>] [ ,P]]
```

```
[<sdir>]MODE LPTn [ : ] =COMn
```

Operation

The MODE command sets the operating parameters of the system video display, serial port, and parallel port. MODE has a different syntax for each device.

Video Display (Options 1 and 2)

MODE performs two operations on the Video Display Adapter; selecting the display mode and aligning it. Each function has its own syntax.

To select the display mode, enter the command line

```
MODE <display>
```

where `display`— is one of the display modes; 40, 80, BW40, BW80, CO40, CO80, or MONO. Each mode is described below.

- | | |
|------|--|
| 40 | This sets the number of characters per line for the Multi-Mode Video Adapter to 40. |
| 80 | This sets the number of characters per line for the Multi-Mode Video Adapter to 80. |
| BW40 | Selects the Multi-Mode Video Adapter to be the active display adapter, disables color or grey scale, and sets the number of characters per line to 40. |
| BW80 | Selects the Multi-Mode Video Adapter to be the active display adapter, disables color or grey scale, and sets the number of characters per line to 80. |
| CO40 | Selects the Multi-Mode Video Adapter to be the active display adapter, enables color or grey scale, and sets the number of characters per line to 40. |
| CO80 | Selects the Multi-Mode Video Adapter to be the active display adapter, enables color or grey scale, and sets the number of characters per line to 80. |
| MONO | Selects a monochrome display adapter to be the active display adapter. Monochrome display adapters only support 80 characters per line. |

In addition to setting the display characteristics, MODE also provides a means for aligning the display on the screen. The display may be adjusted left or right on the screen (one character at a time in 40 column mode and two in 80 column) and MODE can generate a test pattern. The following command line sets the display to Color/80 column, shifts the display right 2 characters, and displays a test pattern.

```
MODE COB0,R,T
```

To shift the display left, enter an "L" in place of the "R". If the "T" option is selected, MODE will ask (after each right or left shift) if you want further adjustment. Enter "Y" to continue shifting the display in the same direction, or "N" to quit.

Serial Communication (Option 3)

The MODE command is used to change the parameters and protocol of the Serial (sometimes called RS-232) port on the Serial/Parallel Adapter. The system supports two Serial ports, 1 and 2. These are referred to as COM1 and COM2.

The following parameters may be specified with MODE. They relate to serial communication protocols.

Baud	This parameter determines the baud rate the serial port will operate at. It may be set to 110, 150, 300, 600, 1200, 2400, 4800, or 9600. When entering this parameter, only the first two digits need be entered; the others are ignored by MODE. For example, 12 will select 1200 baud operation. The default setting is 2400 baud.
Parity	This parameter determines the parity used. It may be set to "N" for no parity, "O" for Odd parity, or "E" for Even parity. "E" is the default.

Databits	This determines the number of databits in each character transmitted. It may be set to either "7" or "8". Seven databits is the default.
Stopbits	This determines the number of stopbits in each character transmitted. It may be set to either "1" or "2". The default is 2 if a baud rate of 110 has been selected and 1 for all others.

The following command line will set the Serial port, COM1, to 1200 baud, Even parity, 8 Databits, and 1 Stopbit.

```
MODE COM1:12,E,8,1
```

Note that only the first two digits of the baud rate have to be entered.

MODE supports one additional option for the Serial port; the "P" option. This indicates that the Serial port is being used as the system printer port, and therefore all time-outs must be continuously retried. The following command line sets the parameters as in the previous example, and specifies the "P" option.

```
MODE COM1:12,E,8,1,P
```

Parallel Printer Port (Options 4 and 5)

MS-DOS supports three parallel printer devices: LPT1, LPT2, and LPT3. The MODE command may be used to set certain parameters for each of those devices. The number of characters per line (<horz>) and the number of lines per inch (<vert>) may be specified. The following command sets LPT2 to 132 characters per line and the vertical spacing to 6 lines per inch.

```
MODE LPT2:132,6
```

The values specified for these parameters are limited; the number of characters per line can be either 80 or 132, and the vertical spacing may be either 6 or 8 lines per inch. Any values entered other than these will be ignored by MODE and the current settings preserved.

The "P" option is also supported for the parallel printer ports by MODE. Specifying this option will instruct MS-DOS to continuously retry all timeouts. The following command line sets LPT2 to the spacing values in the previous example, and enables the "P" option.

```
MODE LPT1:132,6,P
```

The final MODE syntax variation allows characters sent by programs or utilities to the printer to be sent to one of the Serial ports (COM1: or COM2:) instead of the parallel printer port. This allows printers which communicate with the system over a Serial communications link to serve as the printer. The following command line will send characters originally intended for LPT1 to COM1.

```
MODE LPT1:=COM1
```

Notes

1. The "P" option does not actually change printer output to that device; it only prepares it for being a printer output device. The output device must be explicitly changed using the `MODE LPTn=COMn` command.
2. A continuous retry loop may be broken by entering `CTRL-Break` from the keyboard. To disable the "P" option for a device, re-enter the parameters, but without the "P" option. For example,

```
MODE LPT1:132,6,P
```

enables the "P" option for LPT1 and

MODE LPT1:132,6

disables it.

3. If the "P" option or the LPTn=COMn syntax are entered, MODE will load resident code into memory. This will increase the size of MS-DOS. The resident portion will remain in memory until MS-DOS is rebooted.

Related Commands

PRINT

MORE

Purpose

The MORE command reads data from the standard input device and prints it to the standard output device one screen at a time.

Type

MORE is an EXTERNAL command.

Syntax

```
[<mdir>]MORE
```



Operation

MORE is one of three filters provided with the MS-DOS operating system. For a detailed explanation of filters and I/O redirection, refer to the chapter titled *Standard Input and Output*. MORE will print a screenful (24 lines) of data, then print the message —More— and pause. It will wait until any key is pressed and then print another screen. This process will continue until the standard input device signals end-of-file.

Related Commands

SORT, FIND

PATH

Purpose

The PATH command sets an alternative directory search path for MS-DOS.

Type

PATH is an INTERNAL command.

Syntax

```
PATH [[<d>:]<path>[[;[<d>:]<path>]...]]
```

Operation

The PATH command provides paths to additional directories for MS-DOS to search if it is unable to find a command, program, or batch file in the current directory. One or more additional search paths may be specified with PATH. The paths are listed in the order in which they will be searched. The following command line will set the search path to C:\SUB1 followed by A:\SUB2.

```
PATH C:\SUB1;A:\SUB2
```

PATH may also be used to cancel an existing search path. The following command line accomplishes this.

```
PATH ;
```

If the PATH keyword is entered without parameters, the current search path is displayed.

Notes

- 1.** An invalid search path will not be detected until MS-DOS attempts to search the alternate directories. If an invalid pathname (either incorrect syntax or a non-existent subdirectory) is encountered, it will be ignored and the search will continue with the next path.
- 2.** PATH will not allow a command or program to implicitly reference data files or program overlays. Only the initial program file, with an extension of .COM, .EXE, or .BAT, will be searched for.
- 3.** Any PATH command is stored in the MS-DOS Environment string.

Related Commands

SET

PRINT

Purpose

The PRINT command prints a file(s) to the printer while you are performing other tasks on your system (often referred to as "background printing").

Type

PRINT is an EXTERNAL command.

Syntax

```
[<sdir>]PRINT[[<dir>][<file name>]...]
          [/D:<device>][/B:<buffsize>]
          [/U:<busyticks>][/M:<maxticks>]
          [/S:<timeslice>]
          [/Q:<queuesize>][/C][/T][/P]
```

Operation

The PRINT command creates and maintains a print queue. A print queue is a list of files which are to be printed as a background task. PRINT provides a wide range of options which control the print queue or the queue's system parameters.

Files may be added to the queue by entering the name of the file in the command line. For example, if you want to queue the files CHPT1.TXT and CHPT2.TXT for printing, enter the command line

```
PRINT CHPT1.TXT CHPT2.TXT
```

PRINT will add the files to the end of the print queue in the order they are listed in the command line.

If the PRINT command is being executed for the first time since the system was reset, and the /D option (see below) wasn't specified, PRINT will ask

Name of list device [PRN]:



If you press the key, the PRN device is used (it is the default). Alternatively, the name of any character device attached to your system may be entered (i.e. LPT1, LPT2, LPT3, PRN, COM1, COM2, AUX, etc.).

PRINT will install a "resident" section of code which performs the background printing. PRINT will display the message

Resident part of PRINT installed

the first time PRINT is run after the system is reset. If PRINT is run without a filename or option in the command line, a list of the files currently in the print queue will be displayed on the screen.

PRINT provides three options which control the print queue and the files in it. These are listed below:

- /C Enter Cancel mode. This option removes a file from the the print queue. The preceding and all following files in the command line are removed from the queue, until either the end of the command line or a /P option (see below) is encountered.
- /T Terminate print queue. This option may be entered to delete all files from the print queue.
- /P Enter print mode. This option inserts files into the print queue. The preceding and all following files in the command line are added to the queue, until either the end of the command line or a /C option is encountered.

PRINT is either in the Print or Cancel mode. The Print mode is the default, and the /P option need only be specified if the Cancel mode has been selected with the /C option.

PRINT provides six options for control of the queue's system parameters. These options may only be specified the first time PRINT is run. These options determine the parameters for the resident portion of PRINT, which is only loaded the first time the PRINT command is run.

- | | |
|------------------|--|
| /D: <device > | Device. Select the output device. Any valid device attached to your system may be specified. |
| /B: <buffsize > | Buffer size. This option sets the size in bytes of the internal buffer. Increasing the value of this buffer may speed up the Print command. The default value is 512 bytes. |
| /U: <busyticks > | Busy Ticks. Specifies the number computer clock ticks that PRINT will wait for until the printer is available. If PRINT waits longer than this value, it gives up its timeslice. The default value is 1. |
| /M: <maxticks > | Specifies how many clock ticks PRINT can have to print a character to the printer. The default value is 2, and maxticks can range from 1 to 255. |
| /S: <timeslice > | Time Slice. Specifies the time slice value. The default value is 8 and the range is 1 to 255. |

/Q:<queuesize> Queue Size. Specifies the number of files allowed in the print queue. The default is 10 and the range is 4 to 32.

Notes

1. The printer cannot be used by any other program while PRINT is printing files. If another program attempts to use the printer, an "Out Of Paper" error will be returned to that program.
2. The disc(s) containing the files listed in the print queue must remain in the system until the print queue is empty (the queue only holds the filenames, not the contents of the files). In addition, the files must not be altered or erased while they are in the print queue.
3. If PRINT encounters a disc error while printing a file, printing of that file is terminated, a message is printed on the printer, and PRINT goes to the next file in the queue.

Related Commands

None.

PROMPT

Purpose

The PROMPT command changes the MS-DOS system prompt.

Type

PROMPT is an INTERNAL command.

Syntax

```
PROMPT [<text>]
```

Operation

The PROMPT command allows you to change the MS-DOS system prompt. The default prompt is the active drive letter followed by the ">" character. The new system prompt may consist of any combination of literal text characters and special characters.

The text may be made up of any of the characters permissible in MS-DOS filenames. For example, a "user friendly" system prompt (HELLO) could be installed with the following command line:

```
PROMPT HELLO
```

The letters "HELLO" will be displayed whenever the default system prompt would have been.

In addition to text characters, there are several special characters which allow more than just text strings to be inserted in the system prompt. These characters are all preceded by the "\$" character to differentiate them from text strings. The table below lists these characters and their meanings.

Character	Prompt
\$	The '\$' character
T	The current system time
D	The current system date
P	The current directory
V	The MS-DOS version number
N	The active drive
G	The '>' character
L	The '<' character
B	The ' ' character
Q	The '=' character
H	The backspace character
<u> </u> (Underline)	A Carriage Return/Line Feed sequence (go to the next line)
E	ASCII Escape code (1BH)

The following command line will set the prompt to the current directory followed by the '>' character.

```
PROMPT #P#G
```

Notes

1. Text and special character strings may be mixed. For example the command line

PROMPT TIME = %T%_DATE = %D

will produce the prompt

TIME = (<Current time>)

DATE = (<Current date>)

2. If ANSI terminal support has been installed, the %E (Escape) special character may be used to produce reverse video, cursor positioning, and other video attributes in the prompt.
3. Any PROMPT command is stored in the MS-DOS Environment string.

Related Commands

SET, ANSI.SYS

RECOVER

Purpose

The RECOVER command provides a means to recover data files with defective sectors in them or to recover an entire directory or disc which has a defective sector in the directory entries.

Type

RECOVER is an EXTERNAL command.



Syntax

```
[<sdir>]RECOVER [<dir>]<file name>
```

or

```
[<sdir>]RECOVER <d>:
```

Operation

RECOVER has two modes of operation: recover a file and recover a disc. The first syntax is for recovering a file, the second for a disc.

If a file has a bad sector in it, parts of it may not be accessible via MS-DOS or an application program. RECOVER will read the file sector by sector, skipping over the defective sector(s). The good sectors will be placed in a file of the same name, and the bad sectors will be marked as defective (in the same manner as FORMAT) and MS-DOS will no longer allocate data to the defective sector(s). The following command line will recover the file BAD.DAT.

```
RECOVER B:BAD.DAT
```

RECOVER will prompt

```
Press any key to begin recovery of the  
file(s) on drive B:
```

When RECOVER has recovered all valid sectors, the message

```
nnnnnn of mmmmm bytes recovered
```

where `nnnnnn` is the number of bytes recovered, and `mmmmm` is the original size of the file.

If a directory has a defective sector, the entire disc may be recovered. RECOVER will search the File Allocation Tables (FATs) for chains of allocation units (or clusters). A directory entry is created for each chain found, with the filename `FILEnnnn.REC` where `nnnn` is a sequential number starting with `0001`.

Related Commands

CHKDSK, FORMAT

RENAME

Purpose

The RENAME (or REN) command changes the name of a file.

Type

RENAME is an INTERNAL command.

Syntax

```
RENAME [<dir>] <sfile> <dfile>
```

Operation

The RENAME command changes the name of the file specified by <sfile> to the name <dfile>. The first file reference may have an optional drive designator and /or pathname, but the second may not. The file will remain in the same directory after its name has been changed. The following command line renames the file CHAP10.TXT to PART10.TXT

```
RENAME CHAP10.TXT PART10.TXT
```

RENAME accepts wildcards, allowing more than one file to be renamed at one time. For example, the command line

```
RENAME *.TXT *.LST
```

will change the extension of all files with .TXT to .LST.

Related Commands

None

RESTORE

Purpose

The purpose of the RESTORE command is to restore files to their original discs after they have been copied with the BACKUP program.

Type

RESTORE is an EXTERNAL command.

Syntax

```
[<sdir>]RESTORE <d>: [<dir>][<file name>]  
[ /S ][ /P ]
```

Operation

The RESTORE command can restore selected files, or an entire disc of files, depending on which files were copied using BACKUP.

In most cases, RESTORE will be used to copy the contents of a set of backup discs back to the original source disc. To restore the files onto drive C: from the disc in drive A:, enter the command line

```
RESTORE A: C:
```

RESTORE will prompt you to insert the discs in the same order as they were generated. Unless the /S option (see below) is specified, only those files belonging in the current directory will be restored.

If you wish to only restore certain files, specify the drive which you want to restore from and the filename(s) of the files to be restored. An optional drive designator and pathname may be used with the filename, if the file is to be restored to other than the current directory.

The command line to restore all the text files from drive A: to the subdirectory C:\SUB1 is shown below.

```
RESTORE A: C:\SUB1\*.TXT
```

To add flexibility to RESTORE, two options are supported.

- /S Restore to Subdirectories. This option allows files to be restored to more than one directory.
- /P Prompt. This option prompts you before restoring a file which has been changed since it was copied using BACKUP, or which has the Read-only attribute set. If RESTORE detects either of these situations, you will be prompted before the file is restored. This prevents losing the most recent version of a file by inadvertently restoring an older version.

Notes

1. Files must be restored to the same drive and directory which they were backed up from.
2. If files are being shared, they may not be restored. RESTORE will display an error message if an attempt is made to do so.

Related Commands

BACKUP, COPY, DISKCOPY

RMDIR

Purpose

The RMDIR command is used to remove a subdirectory from a disc.

Type

RMDIR is an INTERNAL command.

Syntax

```
RMDIR [<d>:]<path>
```

Operation

RMDIR is used to remove an unwanted subdirectory from a disc. The directory cannot contain any files or subdirectory entries. If an attempt is made to remove a subdirectory which isn't empty, an error message will be displayed. The following command line will remove the subdirectory SUB1 from drive C:

```
RMDIR C:\SUB1
```

Notes

1. The Root and Current Directories for all drives may not be removed.
2. A directory is not empty if it contains hidden files.

Related Commands

MKDIR, CHDIR

SET

Purpose

The SET command is used to set or display the current MS-DOS environment.

Type

SET is an INTERNAL command.

Syntax

```
SET [<label>=<parameter>]
```

Operation

The SET command is used to add, display, and delete strings from the MS-DOS environment. The MS-DOS environment is available to all programs, but is not used by all programs.

The environment consists of a series of character strings. Each string consists of a label and a parameter separated by an '='. MS-DOS inserts the string into the environment verbatim.

The command line

```
SET FILE=CHPT1.TXT
```

will insert the string FILE=CHPT1.TXT into the environment. A string may be deleted by entering the label with no parameter. To remove the string entered above enter the command line

```
SET FILE=
```

If the SET command uses a label which is already in the environment, the new parameters will replace the old

parameters in the environment; duplicate labels are not allowed. For example, if the string FILE=CHPT1.TXT was in the environment and the command line

```
SET FILE=CHPT2.TXT
```

was entered, the string FILE=CHPT2.TXT would replace the string FILE=CHPT1.TXT in the environment.

The current environment may be displayed using the SET command with no label or parameters in the command line.

```
SET
```

will display the current environment on the screen.

Notes

1. The environment will always contain the string COMSPEC = <path>, where <path> is the path to the command processor COMMAND.COM. In addition, the PATH and PROMPT commands will place strings in the environment when they are executed.
2. The SET command may pass variable names to batch files. Refer to the chapter titled *Batch Processing* for information on how to use this feature of Batch processing.

Related Commands

PATH, PROMPT

SHARE

Purpose

The SHARE command loads support for file sharing.

Type

SHARE is an EXTERNAL command.

Syntax

```
[<mdir>]SHARE [/F:<filesize>][/L:<locks>]
```

Operation

The SHARE command is used to provide file sharing in a networked system. It supports two options which change the default system parameters.

- /F Sets the number of bytes reserved for the filenames of shared files. Each file opened in the share mode requires 11 bytes plus the number of characters in the filename and path. The default value is 2048 bytes.

- /L Sets the number of locks allowed. The default value is 20 locks.

Notes

1. SHARE may only be loaded once. If you attempt to load it again, the error message

Share already installed
is displayed.

Related Commands

None

SORT

Purpose

The SORT command reads data from the standard input device, sorts it, and prints it to the standard output device.

Type

SORT is an EXTERNAL command.



Syntax

```
[<mdir>]SORT [/R][/+<n>]
```

Operation

SORT is one of three filters provided with the MS-DOS operating system. SORT will accept up to 63 Kbytes of input data, sort it, then write it to the standard output device. SORT supports the two options listed below:

/R Sort in Reverse order. This option instructs SORT to reverse the normal sort order, i.e. sort from Z to A.

/+ <n> Start with column n. If this option is not specified, the default column number is 1.

SORT is designed to sort text. Therefore, it works with lines (all characters up to and including the CR/LF sequence). It rearranges the lines such that the characters in column n ascend or descend in alphabetical sequence. If the nth character in two or more lines are the same, SORT will compare the nth+1 character. SORT will continue this until either the characters are different or the end of line (CR/LF) is encountered.

The following command line will take the output from the DIR command, sort it by size (the file size starts in column 14), and display it on the screen.

```
DIR | SORT /+14
```

Notes

1. European characters (80H-AFH) are first translated to U.S. characters before being sorted.
2. Upper- and lower-case is ignored during SORT.

Related Commands

FIND, MORE

SUBST

Purpose

The SUBST command allows a drive designator to be substituted for a path.

Type

SUBST is an EXTERNAL command.

Syntax

```
[<sdir>]SUBST <d>: <d>:<path>
```

```
[<sdir>]SUBST <d>:/D
```

```
[<sdir>]SUBST
```

Operation

The SUBST command is useful for applications which don't recognize paths, or for frequently used subdirectories. The subdirectory specified in the command line becomes the root directory of a "virtual drive".

The following command line substitutes the drive designator E: for the subdirectory C:\LEVEL1\LEVEL2.

```
SUBST E: C:\LEVEL1\LEVEL2
```

All references to drive E: will now use this subdirectory.

A substitution may be cancelled with the following command line. The drive designator specified in the original substitution is used.

```
SUBST E:/D
```

If you wish to find out which substitutions are in effect, type

SUBST

Notes

1. When a drive has been substituted, the subdirectory is still accessible on its "physical" drive. In the example above, the subdirectory may be referenced as drive E: or as C:\LEVEL1\LEVEL2 concurrently.
2. When a drive is substituted, the subdirectory specified by the path in the command line becomes the root directory of that drive. All downward subdirectory relationships remain. For example, if the subdirectory shown in the example above had a subdirectory (C:\LEVEL1\LEVEL2\LEVEL3), it could be referenced with the path, E:\LEVEL3.
3. The substituted drive designator must not be higher than the last drive designator specified with the LASTDRIVE command in the CONFIG.SYS file. If the LASTDRIVE command is not included in the CONFIG.SYS file, the default is drive E:. Attempting to substitute drive F: in this case will produce an error message.
4. The following commands should not reference a substituted drive:

ASSIGN	FORMAT
BACKUP	JOIN
DISKCOMP	LABEL
DISKCOPY	RESTORE
FDISK	

5. SUBST may not be used with a drive designator or subdirectory which has been redirected over the network.
6. If you use the designator of one of the physical drives in your system, that drive will be inaccessible while the substitution is in effect. It is recommended that only drive designators not associated with a physical drive be substituted.
7. The current directory on any drive cannot be substituted.

Related Commands

ASSIGN

SYS

Purpose

The SYS command is used to transfer the hidden MS-DOS files, and optionally PAM, to the specified disc.

Type

SYS in an EXTERNAL command.

Syntax

```
[<sdir>]SYS <d>:[/P]
```

Operation

SYS transfers the two hidden MS-DOS files IO.SYS and MSDOS.SYS to the specified drive. The specified disc must have been formatted with the /S, /P, or /B option (see FORMAT). The two MS-DOS files must be in the root directory of the active drive. The following command line transfers the MS-DOS files to drive C:

```
SYS C:
```

If the /P option is specified, the PAM files are transferred in addition to the two MS-DOS files.

Notes

- 1.** Some application programs are distributed on discs which have been formatted with the /B option. SYS may be used to transfer the MS-DOS files to these discs, converting them into system discs.
- 2.** COMMAND.COM is not transferred by SYS unless /P is specified. COPY must be used to transfer it.
- 3.** If /P is specified, CONFIG.SYS will be transferred.

Related Commands

FORMAT

TIME

Purpose

The TIME command is used to display and set the system time.

Type

TIME is an INTERNAL command.

Syntax

```
TIME [<hh:mm>[:<ss>[<.xx>]]]
```

Operation

The TIME command sets the MS-DOS system clock or displays the current time. In order to set the time, the hours and minutes (hh:mm) must be entered. The seconds and hundredths of a second (ss.xx) are optional; MS-DOS will set them to 00.00 if they aren't entered. The following command line sets the time to 1:30 PM. Note that MS-DOS uses 24-hour time format.

```
TIME 13:30:00.00
```

If the time isn't specified, MS-DOS will display the current system time, then prompt for a new time. You may enter a new system time, or press , which will preserve the current system time.

```
TIME
Current time is 13:30:00.00
Enter new time:
```

Notes

1. The TIME command will check for invalid times (i.e. 30 o'clock, etc.). The error message

`Invalid time`

will be displayed, and you will be prompted to re-enter a correct time.

2. The TIME display format may be modified by the COUNTRY command in the CONFIG.SYS file.
3. The TIME command sets the MS-DOS system time, but doesn't change the time kept by the Real-time clock. To change the Real-time clock's time, refer to the SETUP utility or use PAM. The MS-DOS system time is set from the Real-time clock each time the system is reset.

Related Commands

DATE

TREE

Purpose

TREE displays all of the subdirectories found on a specified drive. It may also be used to display the files in those subdirectories.

Type

TREE is an EXTERNAL command.

Syntax

```
[<mdir>]TREE [<d>:][/F]
```

Operation

The TREE command allows the directory structure of a disc to be displayed. If the /F option is specified, the names of the files within the directories will also be displayed.

The command line

```
TREE C:
```

will display the names of all directories on drive C:. For each directory found, the pathname of the directory, plus the pathname of any subdirectories within it will be displayed.

The information on each directory is displayed in the following format:

```
Path: \SUB1  
  
Subdirectories: None  
  
Files: FILE1.TXT  
       FILE2.TXT
```

The heading and names of the files in the directory are only displayed if the /F option has been specified.

Note

1. The output from TREE can be quite lengthy, particularly if the /F option has been specified. The screen display can be halted for easier reading with the Pause Screen function (`CTRL` `Num lock`). Another way is to pipe the output of TREE through the MORE filter using the command line

```
TREE <d>: [/F] | MORE
```

Related Commands

None

TYPE

Purpose

The TYPE command displays the contents of a file on the screen.

Type

TYPE is an INTERNAL command.

Syntax

```
TYPE [<dir>]<file name>
```

Operation

The TYPE command may be used to examine the contents of a file. TYPE does not modify the file except to expand TAB characters to the nearest 8-character column (i.e. 8, 16, 24, etc.). TYPE is primarily intended to display ASCII text files; program and certain data files contain non-ASCII characters and will produce an unintelligible display. The following command line displays the file CHPT1.TXT

```
TYPE CHPT1.TXT
```

Notes

1. Wildcards may not be used in the filename (i.e. TYPE can only display one file at a time). If wildcards are detected, TYPE will display an error message.
2. The `[PrtSc]` function may be used to print the contents of the file to the printer as the file is displayed on the screen. `[CTRL][Num lock]` may be used to pause the display; any subsequent character will start it again. `[CTRL][Break]` will terminate the display and return to the system prompt.

Related Commands

None

VER

Purpose

The VER command returns the current MS-DOS version number.

Type

VER is an INTERNAL command.

Syntax

```
VER
```

Operation

The VER command displays the current MS-DOS version number. The command line

```
VER
```

will produce the display

```
MS-DOS Version 3.10
```

Related Commands

None

VERIFY

Purpose

The VERIFY command turns on or off the optional disc write verify function of MS-DOS.

Type

VERIFY is an INTERNAL command.

Syntax

```
VERIFY [ON|OFF]
```



Operation

The VERIFY command allows the optional MS-DOS disc write verify function to be turned on or off, or the current status of this option displayed.

This function is off when the system is reset. To turn it on, enter the command line:

```
VERIFY ON
```

From this point on, MS-DOS will perform a verify operation after each disc write operation. This function will remain in effect until the VERIFY command is issued with the OFF parameter.

If you execute the VERIFY command with no parameters, MS-DOS will display the current state of the option (ON or OFF).

Notes

1. The verify function may be turned ON or OFF through a system call by an application program. Thus, it is possible for the option to be ON, even though the VERIFY command has not been executed.

Related Commands

CHKDSK

VOL

Purpose

The VOL command displays the volume label of a disc.

Type

VOL is an INTERNAL command.

Syntax

```
VOL [<d>:]
```

Operation

The VOL command displays the disc volume label of the specified drive. If you do not specify a drive, the active drive is assumed. The following command line displays the volume label for drive C:

```
VOL C:
```

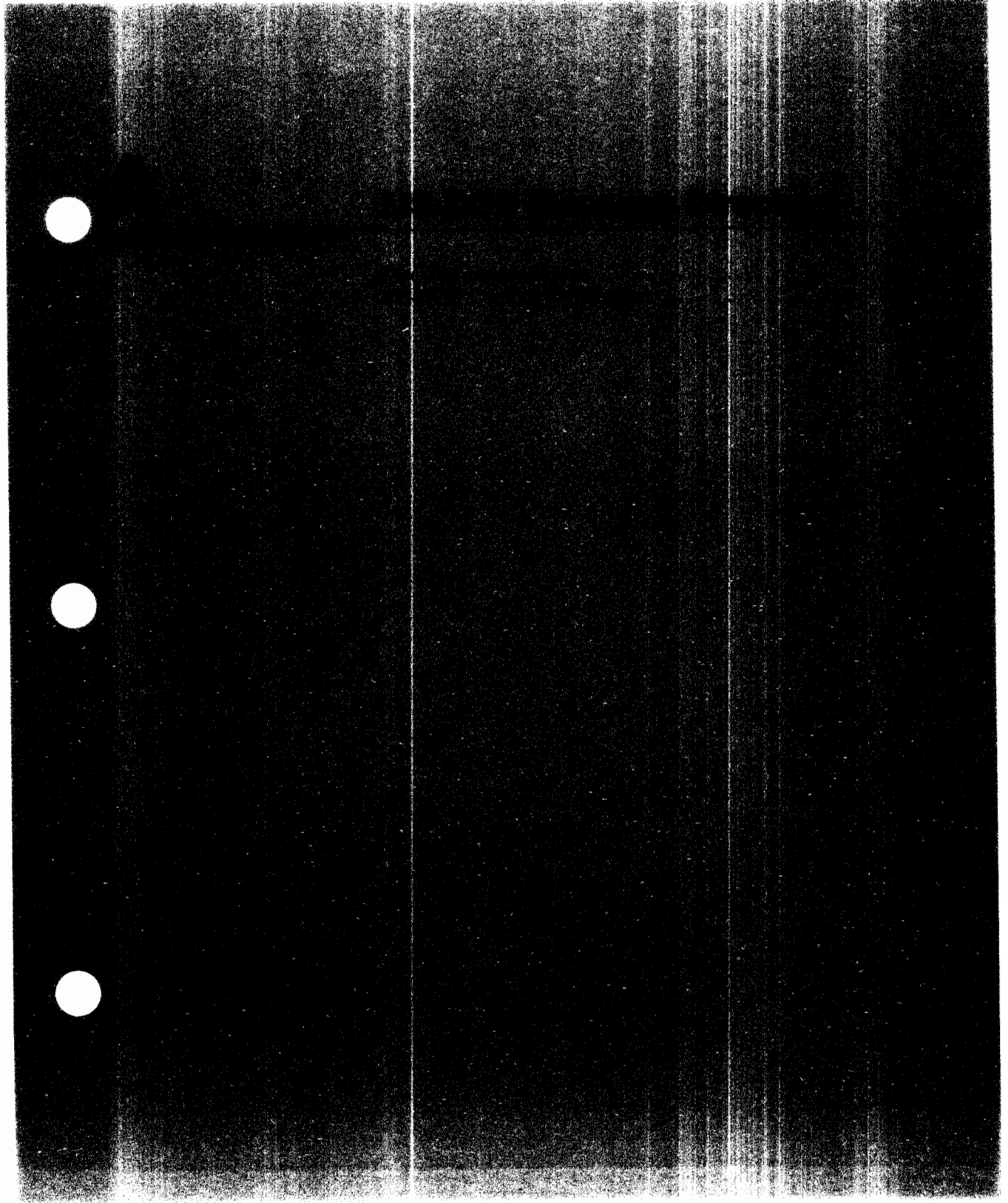
```
Volume in drive C is HARD_DISC
```

If the drive has no label, the following message is displayed.

```
Volume in drive C has no label
```

Related Commands

FORMAT, LABEL



A

MS-DOS Message Directory

MS-DOS messages are issued when MS-DOS or an MS-DOS command encounters an error, needs to inform the user of system status, or prompt the user for an action.

MS-DOS messages are generated by an MS-DOS command, or by the resident portion of MS-DOS. Messages generated by the resident portion of MS-DOS only occur when an error condition is encountered on a disc or device. These disc and device error messages are listed in the next section. The messages generated by the various MS-DOS commands consist of error messages, informational messages, and prompts. They are listed following the disc and device error messages.

Disc and Device Errors

If a disc or device error occurs at any time during a command or program, MS-DOS displays an error message in the following format:

```
<type><action> drive <x>  
Abort, Retry, Ignore?
```

In this message, <type> may be one of the following error conditions:

Write protect error
Bad unit error
Not ready error
Bad command error
Data error
Bad call format error
Seek error
Non-DOS disk error
Sector not found error
No paper error
Write fault error
Read fault error
General failure
Sharing violation
Lock violation
Invalid disk change
FCB unavailable

The <action> may be either READING or WRITING. The drive designator indicates the drive in which the error occurred. For example, if an attempt was made to write to a disc in drive A: which had a write protect tab on it, the error message would be

```
Write protect error writing drive A
Abort, Retry, Ignore?
```

MS-DOS waits for you to respond in one of the following ways:

- | | |
|----------|---|
| A-Abort | End the program requesting the disc read or write. |
| I-Ignore | Ignore the error condition and proceed with the program. |
| R-Retry | Repeat the operation. This response should be used if you have corrected the error (such as removing the write protect tab in the example above). |

One additional error message that might be encountered is

FILE ALLOCATION TABLE BAD FOR DRIVE x

This message means that the copy in memory of one of the allocation tables has error in it. If this error is encountered, CHKDSK should be run with the /F parameter specified. If this error persists, the disc is unusable and must be formatted prior to further use.

MS-DOS Command Messages



MESSAGE: Abort Edit (Y/N)?

Command: EDLIN

Cause: This message is displayed when you enter the Q (Quit) command in EDLIN. The Quit command exits the editing session without saving any editing changes.

Remedy: Enter "Y" if you wish to exit without saving the editing changes to disc, or "N" if you wish to cancel the command and continue the editing session. Use the "E" command to exit EDLIN and save editing changes.

MESSAGE: Abort, Retry, Ignore?

Command: MS-DOS

Cause: A disc or disc device error has been detected by MS-DOS during a program or command's execution.

Remedy: Enter "A" if you wish to abort the program, "R" if you wish to have MS-DOS attempt the operation again, or "I" if you want MS-DOS to proceed with the program or command. "I" is not recommended.

MESSAGE: ABORTED - Assigned Port # missing or invalid

Command: MODE

Cause: This error message is displayed when printer redirection is specified in the command line and the serial port specified was not present in the system or the port specified was not COM1: or COM2:.

Remedy: Re-enter command line.

MESSAGE: ABORTED - Bad device parameter keyword

Command: MODE

Cause: This error message is displayed when a device other than LPTn: or COMn: is specified in the command. MODE only affects the parallel ports, serial ports, and screen.

Remedy: Re-enter command line.

MESSAGE: ABORTED - Illegal BUSY value

Command: MODE

Cause: This error message is displayed when something other than "P" is specified to initialize a port as the printer device.

Remedy: Re-enter command line.

MESSAGE: ABORTED - Illegal STOPBITS value

Command: MODE

Cause: This error message is displayed when a value of other than 1 or 2 is specified for the number of stop bits.

Remedy: Re-enter command line.

MESSAGE: ABORTED - Illegal DATABITS value

Command: MODE

Cause: This error message is displayed when a value of other than 7 or 8 is specified for the number of data bits.

Remedy: Re-enter command line.

MESSAGE: ABORTED - Invalid Switch

Command: MODE

Cause: This error message is displayed when an invalid parameter was given in the command line.

Remedy: Re-enter command line.

MESSAGE: All files cancelled by operator

Command: PRINT

Cause: This is an informational message displayed by PRINT when the /T (terminate queue) option has been specified.

Remedy: None.

MESSAGE: All specified files are contiguous

Command: CHKDSK

Cause: This is an informational message displayed by CHKDSK. It indicates that all files are allocated contiguously on the disc and that there is no fragmentation.

Remedy: None.

MESSAGE: Allocation error in file, size adjusted

Command: CHKDSK

Cause: This message is displayed by CHKDSK when the size of the file indicated in the directory is not consistent with the amount of data actually allocated to the file. The name of the file in error precedes this message.

Remedy: If the /F option was specified for CHKDSK, the file is truncated at the end of the last valid cluster.

If the /F option was not specified, this is an informational message only, and no corrective action has been taken by CHKDSK. CHKDSK must be re-run with the /F option specified to correct the error.

MESSAGE: Are you sure (Y/N)?

Command: DEL,ERASE

Cause: This cautionary message is displayed if you instruct MS-DOS to delete all of the files in a directory.

Remedy: Enter "Y" if you want to proceed or "N" if not.

MESSAGE: Bad command or file name

Command: MS-DOS

Cause: This message is displayed when MS-DOS is unable to find the program or command just entered.

Remedy: Re-enter the command line with the correct spelling of the program or command, move a copy of the program or command file into an accessible directory, change the current directory to one that contains the file, or issue a PATH command to instruct MS-DOS where to look for the command.

MESSAGE: Bad file

Command: FC

Cause: The message is displayed when FC finds a defect in one of the files specified.

Remedy: Run CHKDSK to verify the integrity of your disc.

MESSAGE: Bad or missing (filename)

Command: MS-DOS

Cause: This message is displayed when MS-DOS can't find the file containing the device driver (when its specified in the file CONFIG.SYS) in the root directory of the boot disc, or an error is detected in the driver as it is being loaded. The name of driver (filename) is displayed in the error message. In either case, the device driver is not incorporated into MS-DOS.

Remedy: Check the spelling of the filename in CONFIG.SYS or copy the file containing the device driver into the root directory of the boot disc.

MESSAGE: Bad or missing Command Interpreter

Command: MS-DOS

Cause: This message is displayed when MS-DOS can't find the file COMMAND.COM (or the file containing the alternate command processor specified in the CONFIG.SYS file using the SHELL command) in the root

directory of the boot disc or if an error was encountered as the file was being loaded.

Remedy: Copy the file containing the command processor into the root directory of the boot disc or check the spelling of the SHELL filename in the CONFIG.SYS file.

MESSAGE: BREAK is off [or on]

Command: MS-DOS

Cause: This is an informational message displayed by MS-DOS to inform you of the current setting of the extended `CTRL-Break` and `CTRL-C` checking.

Remedy: None.

MESSAGE: Buffer size adjusted

Command: VDISK

Cause: This informational message is displayed when it is necessary to adjust the buffer size specified in the `DEVICE=VDISK.SYS` command in the CONFIG.SYS file.

Remedy: Change buffer size if necessary.

MESSAGE: Cannot CHDIR to (path) - tree past this point not processed

Command: CHKDSK

Cause: This message is displayed by CHKDSK if it is unable to reach the specified subdirectory. All subdirectories beneath this directory will not be verified.

Remedy: No action required.

MESSAGE: Cannot CHDIR to root. Processing cannot continue

Command: CHKDSK

Cause: This message is displayed by CHKDSK if it is unable to return to the root directory while it is verifying the tree directory structure. CHKDSK will be unable to continue checking the subdirectories in the root.

Remedy: No action required

MESSAGE: Cannot CHKDSK a Network drive

Command: CHKDSK

Cause: This message is displayed by CHKDSK if a drive has been specified that is redirected over the network. CHKDSK can only operate on local disc drives.

Remedy: Do not specify a disc drive that is not local.

MESSAGE: Cannot do binary reads from a device

Command: COPY

Cause: This message is displayed by COPY when an attempt is made to copy a file from a device and the /B option has been specified. This mode is invalid due to the fact that COPY needs to be able to detect the end-of-file character from the device.

Remedy: Re-enter the command line and omit the /B option or specify the /A option after the device name in the command line.

MESSAGE: Cannot edit .BAK file -- rename file

Command: EDLIN

Cause: This error message is displayed by EDLIN when the filename specified to edit has an extension of .BAK.

Remedy: Rename the file giving it a different extension, or copy the file to a new file with an extension other than .BAK.

MESSAGE: Cannot format an Assigned drive

Command: FORMAT

Cause: This error message is displayed by FORMAT when a drive with an active assignment has been specified as the target drive.



Remedy: Use the ASSIGN command to clear the assignment and reenter the FORMAT command line.

MESSAGE: Cannot format a Network drive

Command: FORMAT

Cause: This error message is displayed by FORMAT when a drive has been specified that is redirected over the network.

Remedy: Do not specify a drive that is not local.

MESSAGE: Cannot open (filename)

Command: PRINT

Cause: This error message is displayed by PRINT when a specified file cannot be found in an accessible directory.

Remedy: Make sure that the filename was correctly entered in the command line, or move the file into an accessible directory.

MESSAGE: Cannot recover . entry, processing continued

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when the active directory (indicated by the single period) is defective.

Remedy: No action required

MESSAGE: Cannot recover a Network drive

Command: RECOVER

Cause: This error message is displayed by RECOVER when a drive has been specified that is redirected over the network.

Remedy: Do not specify a drive that is not local.

MESSAGE: Cannot recover .. entry

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when the parent directory (indicated by the dual periods) is defective.

Remedy: No action required

MESSAGE: Compare error on side x, track y

Command: DISKCOMP

Cause: There are one or more locations on the indicated side and track of the two discs that contain differing information.

MESSAGE: Content of destination lost before copy

Command: COPY

Cause: This error message is displayed by COPY when the same file has been specified as both a source and the destination drive. The following command line is an example of this situation

```
COPY FILE1+FILE2 FILE1
```

The file FILE1 is overwritten before it can be copied.

Remedy: Copy to a temporary file, delete the duplicate file, and rename the temporary file to the desired filename.

MESSAGE: Convert lost chains to files (Y/N)?

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when it detects lost clusters or chains (multiple clusters).

Remedy: If you answer "Y", each chain will be converted to a file with the filename FILEnnnn.CHK, where nnnn is a sequential number starting with 0000. If you answer "N", CHKDSK frees the lost chains, and their space may be reallocated by MS-DOS. These actions only occur if the /F option has been specified. If not, CHKDSK will ask the question, but no action will be taken, regardless of the answer.

MESSAGE: Copy Another (Y/N)?

Command: DISKCOPY

Cause: This message is displayed by DISKCOPY when it has completed copying a disc.

Remedy: Answer "Y" if you want to copy another disc, "N" if not.

MESSAGE: Copy complete

Command: DISKCOPY

Cause: This is an informational message displayed by DISKCOPY when it has completed copying a disc.

Remedy: None required.

MESSAGE: Copy not completed

Command: DISKCOPY

Cause: This error message is displayed by DISKCOPY when it can't copy the entire disc. This error may be due to a defect on either the source or destination disc.

Remedy: If the error is on the destination disc, use a new disc which has no defects. If the error is on the source disc, use COPY to copy the files to another disc.

MESSAGE: Copying...

Command: DISKCOPY

Cause: This informational message is displayed by DISKCOPY to indicate that it is copying a disc.

Remedy: No action required.

MESSAGE: Corrections will not be written to disk

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when an error(s) has been detected on the disc, but the /F option has not been specified.

Remedy: Run CHKDSK again with the /F option specified.

MESSAGE: Data left in [filename]

Command: FC

Cause: This informational message is displayed by FC if there is data left on one of the files being compared after the end of the file is reached in the other.

Remedy: No action required.

MESSAGE: Directory entries adjusted

Command: VDISK

Cause: This informational message is displayed

when it is necessary to adjust the number of directory entries specified in the DEVICE=VDISK.SYS command in the CONFIG.SYS file.

Remedy: Change number of directory entries if necessary.

MESSAGE: Directory is joined

Command: CHKDSK

Cause: This error message is displayed by CHKDSK if a directory on the disc being checked has been joined.

Remedy: Unjoin the directory and run CHKDSK again.

MESSAGE: Directory not empty

Command: JOIN

Cause: This error message is issued by JOIN if an attempt is made to join to a directory which is not empty.

Remedy: Join to a new directory which is empty, or empty the directory and rerun JOIN.

MESSAGE: Directory is totally empty, no . or ..

Command: CHKDSK

Cause: This error message is displayed by

CHKDSK if a subdirectory does not contain the parent or current directory entries.

Remedy: Delete the subdirectory with RMDIR and recreate it using MKDIR.

MESSAGE: Disk error reading FAT [x:]

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when one of the two File Allocation Tables for drive x: has a defective sector in it. MS-DOS will automatically use the good FAT.

Remedy: Copy the files to another disc and reformat the disc. If the sectors for the FAT are still defective, discard the disc.

MESSAGE: Disk error writing FAT [x:]

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when one of the two File Allocation Tables for drive x: has a defective sector in it. MS-DOS will automatically use the good FAT.

Remedy: Copy the files to another disc and reformat the disc. If the sectors for the FAT are still defective, discard the disc.

MESSAGE: Disk full. Edits lost

Command: EDLIN

Cause: This error message is displayed by EDLIN whenever it has not able to save your file due to lack of disc space.

Remedy: Delete enough files to make room for your document and re-enter it with EDLIN.

MESSAGE: Disk full--write not completed

Command: EDLIN

Cause: This error message is displayed by EDLIN if it was not able to save your file due to lack of disc space.

Remedy: Delete enough files to make room for your document and re-enter it with EDLIN.

MESSAGE: Disk unsuitable for system drive

Command: FORMAT

Cause: This error message is displayed by FORMAT if it detects a bad track on the disc where the system files should reside.

Remedy: This disc should only be used for data. Use another disc if you want to make a system disc.

MESSAGE: Diskette/Drive not compatible

Command: DISKCOMP

Cause: Disks which are being compared are of differing densities.

Remedy: Only compare disks of similar densities.

MESSAGE: Diskettes compare ok

Command: DISKCOMP

Cause: This informational message indicates the two diskettes contain identical information.

Remedy: No action required

MESSAGE: Disks must be the same size.

Command: DISKCOPY

Cause: This error message is displayed by DISKCOPY if you attempt to use DISKCOPY with two discs with different format and capacities.

Remedy: Use discs with the same format or use the COPY command to transfer files and SYS to transfer the system files (if desired).

MESSAGE: Divide overflow

Command: MS-DOS

Cause: This error message is displayed by MS-DOS if a program attempts to divide by 0, or a logic error caused an internal malfunction.

Remedy: Correct error in program. If the program is purchased, contact your dealer.

MESSAGE: [.] [..] Does not exist

Command: CHKDSK

Cause: This error message is displayed by CHKDSK if either the . or .. entry in a subdirectory is invalid.

Remedy: No action required

MESSAGE: Copy Process Ended

Command: DISKCOPY

Cause: This error message is displayed when trying to copy discs of differing densities.

Remedy: Only copy disks of similar densities.

MESSAGE: Drive x: not ready.

Remedy: Make sure a diskette is inserted into the drive and the door is closed.

MESSAGE: Press any key when ready...

Command: DISKCOMP, DISKCOPY

Cause: One of the disc drives was not ready.

Remedy: If there is a disc in the drive, open the door, reinsert the disc, and reclose the door.

MESSAGE: Duplicate file name or File not found

Command: RENAME

Cause: Either the file being renamed does not exist, or the new file name already exists.

Remedy: Be sure the file being renamed exists, and that the new file name does not exist.

MESSAGE: End of input file

Command: EDLIN

Cause: This informational message is displayed by EDLIN when the entire file has been read into memory. If the file has been read in sections (using the Append command) this message is displayed when the last section is read.

Remedy: No action required

MESSAGE: EOF mark not found

Command: COMP

Cause: This informational message is displayed

by COMP when one or both of the file being compared do not contain End-of-file markers.

Remedy: No action required



MESSAGE: Entry error

Command: EDLIN

Cause: This error message is displayed by EDLIN when it detects a syntax error in a command.

Remedy: Retype the command with the correct syntax.

MESSAGE: Entry has a bad [attribute or link or size]

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when one of these error conditions is detected in one of the subdirectory entries. The message will be preceded by one or two periods to indicate which entry.

Remedy: None. CHKDSK will attempt to correct the error if the /F option has been specified.

MESSAGE: Error in .EXE file

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when an .EXE program is attempting to

load and execute an invalid internal format.

Remedy: Relink program or make a new copy to execute. If you are using a purchased program and error persists, contact your dealer.

MESSAGE: Error writing to device

Command: MS-DOS

Cause: Too much data was sent to a device. MS-DOS was unable to write the data to the specified device.

Remedy: Send less data to the device.

MESSAGE: Errors found, F parameter not specified. Corrections will not be written to disk.

Command: CHKDSK

Cause: This message is displayed by CHKDSK when it detects an error and the /F option has not been specified. Any corrective action taken by CHKDSK will not be written to disc.

Remedy: Rerun CHKDSK with the /F option specified in order to correct the errors.

MESSAGE: Errors on list device indicate that it may be off-line. Please check it.

Command: PRINT

Cause: This message is displayed by PRINT if a printer time-out error is detected. This error may also be displayed during a long printer operation such as a form feed. This message is only displayed when the PRINT command is executed.

Remedy: Turn the printer on or return it to on-line status if appropriate. If error occurs during a long printer operation, ignore the message.

MESSAGE: EXEC failure

Command: MS-DOS

Cause: This error message is displayed by MS-DOS either when an error is found while reading a command file, or when the FILES command in the CONFIG.SYS file is set too low.

Remedy: Increase the FILES value and restart MS-DOS, or check the integrity of the disc with the command file.

MESSAGE: File allocation table bad

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when a disc may be defective.

Remedy: Run CHKDSK to check the disc.

MESSAGE: File allocation table bad drive (x:)

Command: CHKDSK, PRINT

Cause: This error message is displayed by CHKDSK or PRINT when a disc may be defective.

Remedy: Run CHKDSK to check disc.

MESSAGE: File cannot be copied onto itself

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when the source filename you specified is the same as the destination filename.
Example:

```
copy a a
```

MESSAGE: File cannot be converted

Command: EXE2BIN

Cause: This error message is displayed by EXE2BIN when the input file is not in the correct format.

MESSAGE: File creation error

Command: MS-DOS

Cause: This error message is displayed by MS-DOS if you try to add a new filename or replace a file that already exists in the directory. If

the file already exists, it is a read-only file and cannot be replaced.

Remedy: Run CHKDSK on the disc to determine the cause.

MESSAGE: File is READ-ONLY

Command: EDLIN

Cause: This error message is displayed by EDLIN if you try to change a file that is designated read-only.

MESSAGE: (filename) cancelled by operator

Command: PRINT

Cause: This message is printed on the printer when you specify the /T switch in the PRINT command.

MESSAGE: (filename) contains non-contiguous blocks

Command: CHKDSK

Cause: This message is displayed by CHKDSK when the filename specified is not allocated contiguously on the disc.

Remedy: No action required

MESSAGE: (filename) file not found

Command: PRINT

Cause: This error message is displayed by PRINT if disks are switched while a file is queued up, but before it started to print.

Remedy: Reissue the PRINT command for that filename.

MESSAGE: (filename) is cross linked on cluster

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when two or more files are cross linked.

Remedy: Make a copy of the file you want to keep, and then delete files that are cross linked.

MESSAGE: (filename) is currently being printed

Command: PRINT

Cause: This message is displayed by PRINT when the filename specified is being printed.

Remedy: No action required

MESSAGE: (filename) is in queue

Command: PRINT

Cause: This message is displayed by PRINT when the filename specified is waiting to be printed.

Remedy: No action required

MESSAGE: Filename must be specified

Command: EDLIN

Cause: This error message is displayed by EDLIN when you did not specify a filename at the time you started EDLIN.

Remedy: Specify a file to edit in the command line to start EDLIN.

MESSAGE: File not found

Command: EDLIN, FC, FIND, MS-DOS, RECOVER, EXE2BIN, REN

Cause: This error message is displayed when MS-DOS cannot find the file that you specified.

Remedy: Check to see that the pathname is accurate and that the file exists in the directory you specified.

MESSAGE: Files are different

Command: FC

Cause: After FC has detected many differences, it gives up and displays this message.

Remedy: Edit the files to remove the reported differences and try again.

MESSAGE: Files are different sizes

Command: COMP

Cause: This informational message is displayed by COMP when the two files are different sizes. The compare operation is terminated.

Remedy: No action required

MESSAGE: Find: File not found <file name>

Command: FIND

Cause: This error message is displayed by FIND when you have specified a file that doesn't exist.

Remedy: Make sure you have typed the filename correctly.

MESSAGE: Find: Invalid number of parameters

Command: FIND

Cause: This error message is displayed by FIND if you have specified too many, or not enough options in the command line.

Remedy: Re-enter the command correctly.

MESSAGE: Find: Invalid Parameter

Command: FIND

Cause: This error message is displayed by FIND if one of the switches you have specified is wrong.

Remedy: Re-enter the command line correctly.

MESSAGE: Find: Read error in <filename>

Command: FIND

Cause: This error message is displayed by FIND if the program could not read the specified file.

Remedy: Run CHKDSK to ensure the integrity of the disc and try again.

MESSAGE: Find: Syntax error

Command: FIND

Cause: This error is displayed by FIND if the command has been typed incorrectly.

Remedy: Re-enter the command line correctly.

MESSAGE: First cluster number is invalid, entry truncated

Command: CHKDSK

Cause: This error message is displayed by CHKDSK if the file directory entry contains

an invalid pointer to the data area. If you specified the /F switch, the file is truncated to a zero length file.

Remedy: No action required

MESSAGE: Fixups needed - base segment (hex:)

Command: EXE2BIN

Cause: This message is displayed by EXE2BIN if the source (.EXE) file contained information indicating that a load segment is required for the file.

Remedy: Specify the absolute segment address at which the finished module is to be located.

MESSAGE: For cannot be nested

Command: BATCH COMMAND PROCESSOR

Cause: This error message is displayed by MS-DOS when FOR commands are nested in a batch file.

Remedy: Edit the batch file to remove the nested FOR statements.

MESSAGE: Format failure

Command: FORMAT

Cause: This error message is displayed when MS-DOS could not format the disc. The

message is usually displayed with an explanation as to why MS-DOS could not format the disc.

Remedy: Restart FORMAT with a new disc.

MESSAGE: Formatting While Copying



Command: DISKCOPY

Cause: This information message is displayed when the destination disc is not recognized as a formatted disc. DISKCOPY will format while it copies from the source disc.

Remedy: No action required

MESSAGE: Has invalid cluster, file truncated

Command: SYS

Cause: This error message is displayed by SYS when the system files IO.SYS and MS-DOS occupy more space on the source disc than is available on the destination disc.

Remedy: No action required

MESSAGE: Incorrect DOS version

Command: CHKDSK, EDLIN, FC, FIND, FORMAT, MORE, PRINT, RECOVER, SORT, SYS

Cause: This error message is displayed because

many version 2.00 utilities will not run on other versions of MS-DOS. The utilities listed above will only run under the exact version of MS-DOS for which they were configured.

Remedy: Use the correct versions of these commands for MS-DOS 3.1

MESSAGE: Incorrect number of parameters

Command: JOIN, SUBST

Cause: This error message is displayed by JOIN or SUBST when you have specified too many or too few options in the command line.

Remedy: Re-enter the command line correctly.

MESSAGE: Incorrect parameter

Command: ASSIGN, SHARE

Cause: This error message is displayed by ASSIGN or SHARE when one of the options you specified is wrong.

Remedy: Re-enter the command line correctly.

MESSAGE: Insert diskette for drive (x:) and strike any key when ready

Command: MS-DOS

Cause: This error message is displayed when MS-DOS is copying and formatting. You should insert a disc in the appropriate drive and press any character or number key to begin processing.

MESSAGE: Insert diskette with batch file and press any key when ready

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when the disc containing the batch file you specified is not in the drive you originally specified.

Remedy: Reinsert the disc that contains the batch file in the appropriate drive.

MESSAGE: Insert DOS diskette into drive (x:) and strike any key when ready

Command: FORMAT

Cause: This error message is displayed by FORMAT when you have specified FORMAT /S but the disc in the default drive does not contain MS-DOS system files.

Remedy: Insert a disc with the files IO.SYS and MSDOS.SYS in the drive specified.

MESSAGE: Insert formatted target diskette into drive (x:)

Command: DISKCOPY

Cause: This error message is displayed when DISKCOPY is ready for a disc in the destination drive. You must format the destination disc with the FORMAT program before using DISKCOPY.

Remedy: Format a disc correctly and then restart DISKCOPY.

MESSAGE: Insert new diskette for drive (x:) and strike any key when ready

Command: FORMAT

Cause: This message will be displayed by FORMAT when you should insert a blank disc into the appropriate drive.

Remedy: Press any character or number key to begin formatting. If there is any data on the disc, it will be destroyed by the format process.

MESSAGE: Insert system diskette in drive (x:) and strike any key when ready.

Command: SYS

Cause: This error message is displayed when SYS needs a disc from which to read the IO.SYS and MSDOS.SYS files into memory.

Remedy: Insert the system disc and press any key to start the system copy process.

MESSAGE: Insert target diskette into drive
(x:)

Command: DISKCOPY

Cause: This error message is displayed by DISKCOPY if you are running DISKCOPY and your source and destination drives are the same.

Remedy: Reinsert the destination disc into the specified drive.

MESSAGE: Insufficient disk space

Command: MS-DOS, SORT, EXE2BIN

Cause: This error message is displayed when the disc is full. It does not contain enough room to perform the specified operation.

Remedy: Delete extraneous files from the disc and try again.

MESSAGE: Insufficient memory for system transfer. Processing cannot continue.

Command: FORMAT

Cause: This error message is displayed by FORMAT when the memory configuration is insufficient to transfer the MS-DOS system files IO.SYS and MSDOS.SYS (the /S switch)

Remedy: Re-boot the system with fewer device drivers, virtual discs, or more memory.

MESSAGE: Insufficient room in root directory. Erase files in root and repeat CHKDSK

Command: CHKDSK

Cause: This error message is displayed by CHKDSK. CHKDSK always recovers lost files into the root directory. In this case, your root directory is full.

Remedy: Delete some files in your root directory to make room for the lost files.

MESSAGE: Intermediate file error during pipe

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when the pipe operation makes use of temporary files on the disc which will

automatically be deleted after the piping process is complete. An error has occurred in one of these files.

Remedy: Check the integrity of your disc with CHKDSK. Try again with a new disc.

MESSAGE: Internal error

Command: FC

Cause: This error message displayed by FC indicates an internal logic error in FC.

Remedy: Use COMP to compare your files.

MESSAGE: Invalid characters in volume label

Command: FORMAT

Cause: This error message is displayed by FORMAT when the volume label contains invalid or too many characters.

Remedy: Use the LABEL command to ensure that the volume label contains up to 11 alphanumeric characters.

MESSAGE: Invalid COMMAND.COM. Insert COMMAND.COM disk in default drive and strike any key when ready

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when the program you have just run has used up almost all of memory. MS-DOS must now reload the COMMAND.COM file from disc. However, MS-DOS cannot find COMMAND.COM on the disc or the copy found is invalid.

Remedy: Insert a disc into the default drive which contains a copy of the COMMAND.COM version on the disc with which you started MS-DOS.

MESSAGE: Invalid country code

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when you have specified a country number in your CONFIG.SYS file which is not in the table of countries configured in this implementation of MS-DOS.

Remedy: See the COUNTRY command for a list of valid country codes.

MESSAGE: Invalid drive specification
Specified drive does not exist, or is non-removable

Command: DISKCOPY

Cause: This error message is displayed if a drive designator was specified for a drive that doesn't exist or for a hard disc.

Remedy: Specify only flexible disc drives that are present.

MESSAGE: Invalid baud rate specified

Command: MODE

Cause: This error message is displayed if a baud rate other than 110, 150, 300, 600, 1200, 2400, 4800, or 9600 is specified.

Remedy: Re-enter command line with valid baud rate.

MESSAGE: Invalid switch character

Command: VDISK

Cause: This error message is displayed if the character following the "/" in the command line wasn't "E". VDISK will attempt to install the virtual disc in low memory.

Remedy: Re-enter command line in CONFIG.SYS file.

MESSAGE: Invalid working directory

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when your disc is bad.

Remedy: Replace the disc or make another copy from your backup system disc.

MESSAGE: Invalid date

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when an invalid date has been specified in response to the date prompt when starting MS-DOS.

Remedy: Retype in correct date.

MESSAGE: Invalid device

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when the device specified was not CON, NUL, AUX, or PRN.

Remedy: Use a valid device name.

MESSAGE: Invalid directory

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when the directory specified either does not exist or is invalid.

Remedy: Check to see that you entered the directory name correctly.

MESSAGE: Invalid drive in search path

Command: MS-DOS

Cause: This error message is displayed when an invalid drive has been detected in the PATH string.

Remedy: Issue a new PATH command with no invalid drives in it.

MESSAGE: Invalid drive or filename

Command: EDLIN, RECOVER

Cause: This error message is displayed by EDLIN or RECOVER when a valid drive or a valid file name needs to be specified.

Remedy: Specify a drive which is present.

MESSAGE: Invalid drive specification

Command: CHKDSK, FORMAT, SYS

Cause: This error message is displayed by CHKDSK, FORMAT or SYS when a valid drive needs to be specified.

Remedy: Specify a valid drive ID.

MESSAGE: Invalid drive specification.
Specified drive does not exist, or is non-removable.

Command: DISKCOPY

Cause: This error message is displayed when a drive designator was specified for a drive that doesn't exist or is non-removable.

Remedy: Reissue the command with a drive that exists and is removable.

MESSAGE: Invalid number of parameters

Command: FC, FIND, RECOVER

Cause: This error message is displayed by FC, FIND or RECOVER when there are a wrong number of options in the command line.

Remedy: Reissue the command with the correct number of options.

MESSAGE: Invalid parameter

Command: CHKDSK, EDLIN, FC, FIND, FORMAT, PRINT

Cause: This error message is displayed by CHKDSK, EDLIN, FC, FORMAT or PRINT when one of the switches that are specified is wrong.

Remedy: Check the correct syntax and reissue the command.

MESSAGE: Invalid parameter. Do not specify filename(s). Command Format:
DISKCOMP d: d: [/L][/B]

Command: DISKCOMP

Cause: This error message is displayed when one or more filenames were encountered. DISKCOMP only accepts drive designators.

Remedy: Re-enter command in proper format.

MESSAGE: Invalid parameter. Do not specify filename(s). Command Format:
DISKCOPY d: d: [/L]

Command: DISKCOPY

Cause: This error message is displayed when one or more filenames were encountered. DISKCOPY only accepts drive designators.

Remedy: Re-enter command in proper format.

MESSAGE: Invalid path or file name

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when a valid pathname or filename to the COPY command needs to be specified.

MESSAGE: Invalid path, not directory, or directory not empty

Command: RMDIR

Cause: This error message is displayed by MS-DOS when you are unable to remove the directory requested for one of the specified reasons.

Remedy: Be sure the directory named actually exists and is empty, then reissue the command.

MESSAGE: Invalid sub-directory entry

Command: CHKDSK
Cause: This error message is displayed by CHKDSK when the subdirectory that you specified either does not exist or is invalid.

Remedy: Check to see that you typed the subdirectory name correctly.

MESSAGE: Invalid time

Command: TIME

Cause: This error message is displayed when you have specified an invalid time in response to the TIME prompt.

Remedy: Enter a valid time in the correct (24-hour) format.

MESSAGE: Label not found

Command: MS-DOS batch file processor

Cause: This error message is displayed by MS-DOS when there is a GOTO command to a nonexistent label in a batch file.

Remedy: Edit the batch file so the label exists, or remove the reference to the label.

MESSAGE: Line too long

Command: EDLIN

Cause: This error message is displayed by EDLIN

when during a Replace command the string given as the replacement caused the line to expand beyond 253 characters.

Remedy: Divide the long line into two lines and retry the Replace command.

MESSAGE: List output is not assigned to a device

Command: PRINT

Cause: This error message is displayed by PRINT. When you first run PRINT, it asks you what device you want to specify as a print spooler. This message appears if PRINT is set up for a device that does not exist.

Remedy: Specify a device name that exists in your system.

MESSAGE: Memory allocation error

Command: MS-DOS



Cause: A program has written into "free" memory, preventing MS-DOS from allocating that memory. This error is usually fatal.

Remedy: When this message is displayed you should restart MS-DOS. If this error persists, make a new copy of the MS-DOS disc from your backup copy of the system disc. If this error occurs consistently with purchased software, see your dealer.

MESSAGE: --More--

Command: MORE

Cause: The program which has piped its output to MORE produces more than one screen of data for MORE to print.

Remedy: When this message is displayed by MORE you should press the spacebar to view more of the file or directory.

MESSAGE: MORE: Incorrect DOS version

Command: MORE

Cause: This error message is displayed because More will not run on versions of MS-DOS previous to 2.0

Remedy: Use the correct version of MORE and MS-DOS.

MESSAGE: Must specify destination line number

Command: EDLIN

Cause: This error message is displayed by More when you must specify a destination line number when you are copying and inserting lines with EDLIN.

Remedy: Reissue the command, including the destination line number this time.

MESSAGE: Must specify 0N or 0FF

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when the command requires either an ON or an OFF argument.

Remedy: Reissue the command, specifying either ON or OFF.

MESSAGE: Name of list device (PRN):

Command: PRINT

Cause: This prompt appears the first time that PRINT is run. Any valid device may be specified and that device then becomes the PRINT output device.

Remedy: Specify the device you want to print on.

MESSAGE: New file

Command: EDLIN

Cause: This message is printed if EDLIN does not find a file with the name you specified.

Remedy: If you are creating a new file, ignore this message. If you do not intend to create a new file, check to see that you correctly typed the filename of the file you wish to edit.

MESSAGE: No files match d:xxxxxxx.xxx

Command: PRINT

Cause: This error message is displayed by PRINT

when a filename was given for files to add to the queue, but no files match the specification.

Remedy: Make sure you typed the name of the files correctly, and reissue the command.

MESSAGE: No free file handles. Cannot start
COMMAND.COM, exiting

Command: MS-DOS

Cause: COMMAND.COM has been unable to open a file because there are not enough file handles left available in the system.

Remedy: Re-start MS-DOS, then add or increase the number in the FILES statement in CONFIG.SYS, then re-start MS-DOS again.

MESSAGE: No path

Command: PATH

Cause: There is no PATH variable set in the environment.

Remedy: No action required

MESSAGE: No room for system on destination disc

Command: SYS

Cause: This error message is displayed by SYS

when there is not enough room for the system files on the destination disc.

Remedy: Delete some files to make room for the system files or use another disc. You may need to reformat the disc to put the system on it.

MESSAGE: No room in directory for file

Command: EDLIN

Cause: This error message is displayed by EDLIN when you have tried to save a file to the root directory but it is full. Subdirectories are not limited in size as is the root directory.

Remedy: Delete extraneous files from the root, or specify a file name in a subdirectory. Re-enter your text into EDLIN.

MESSAGE: Not enough memory

Command: JOIN, SHARE, SUBST

Cause: This error message is displayed by JOIN, SHARE, or SUBST when there is not enough memory for MS-DOS to run the command.

Remedy: Reduce the number of installed drivers and virtual discs, or acquire more memory for your system.

MESSAGE: Not enough room to merge the entire file

Command: EDLIN

Cause: This error message is displayed by EDLIN if there is not enough room in memory to hold the file during a Transfer command.

Remedy: You must free some memory by writing some of the text to disc before reissuing the Transfer command.

MESSAGE: Not found

Command: EDLIN

Cause: This error message is displayed by EDLIN if you have specified a Search or a Replace command that was unable to find a further occurrence of the specified Search or Replace string.

Command: EDLIN

Remedy: No action required

MESSAGE: 0 . K . ?

Command: EDLIN

Cause: This prompt occurs during Search and Replace command processing.

Remedy: If you press any key except Y or the key, the search or replace process continues.

MESSAGE: Out of environment space

Command: SET, PATH, PROMPT

Cause: This error message is displayed by MS-DOS when there is not enough room in the program environment to accept more data.

Remedy: Remove unnecessary strings from the environment. Reissue the command.

MESSAGE: Path not found

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when you have specified an invalid pathname.

Remedy: Check to be sure you have entered the pathname correctly and that the path exists.

MESSAGE: Port not installed

Command: MODE

Cause: This error message is displayed if the port specified (LPTn: or COMn:) is not present in the system.

Remedy: Specify a port that is in the system.

MESSAGE: Press any key to begin formatting (x:)

Command: FORMAT

Cause: This prompt is issued before you format a disc.

Remedy: Press any character or number to begin the format process. If you wish to end this command, press `CTRL C`

MESSAGE: Press any key to begin recovery of the (xxx) file(s) on drive (x:)

Command: RECOVER

Cause: This prompt is issued before you recover a disc or file.

Remedy: Press any character or number key to begin the recover process. If you wish to end this command, press `CTRL C`

MESSAGE: PRINT queue is empty

Command: PRINT

Cause: This message is displayed by PRINT when there are no files waiting to be printed.

Remedy: No action required

MESSAGE: PRINT queue is full

Command: PRINT

Cause: This error message is displayed by PRINT when there is no room in the list of files waiting to be printed.

Remedy: Wait until some of the waiting files have printed, or remove them from the queue. Reissue the PRINT command.

MESSAGE: Probable non-DOS disk Continue (Y/N)?

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when the disc you are using is not recognized by this version of MS-DOS. The disc either was created by another system with a format that is not supported on this version of MS-DOS or is not an MS-DOS disc.

Remedy: If the /F parameter was used, reissue the command without /F. The possible corrections will then be displayed. The disc may then be recovered with /F, or reformatted.

MESSAGE: Processing cannot continue



Command: CHKDSK

Cause: This error message is displayed by CHKDSK when there is not enough memory in your machine to process CHKDSK for this disc.

Remedy: Remove some drivers or virtual discs, or obtain more memory for your system.

MESSAGE: Program too big to fit in memory

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when you must acquire more memory to run your application. It is possible that some programs you have run are still using some memory.

Remedy: You may try to restart MS-DOS; however, if you still receive this message, you must acquire more memory.

MESSAGE: Read error in (filename)

Command: FC, FIND

Cause: This error message is displayed by FC or FIND when MS-DOS could not read the file.

Remedy: Run CHKDSK to ensure the integrity of the files. RECOVER them if necessary.

MESSAGE: Resident part of PRINT installed

Command: PRINT

Cause: This is the first message that MS-DOS displays when you issue the PRINT command. It means that available memory has been reduced by several thousand bytes to process the print command concurrent with other processes.

Remedy: No action required

MESSAGE: Reinsert diskette for drive x

Command: FORMAT

Cause: This message is displayed by FORMAT when you should reinsert the disc being formatted in the indicated drive.

Remedy: Insert the disc to be formatted in the specified drive.

MESSAGE: Sector size adjusted

Command: VDISK

Cause: This informational message is displayed when it is necessary to adjust the sector size specified in the DEVICE=VDISK.SYS command in the CONFIG.SYS file.

Remedy: Change sector size if necessary.

MESSAGE: Sector size too large in file
(filename)

Command: MS-DOS

Cause: This error message is displayed by MS-DOS when the specified device driver loaded by CONFIG.SYS uses a sector size larger than that of any other device driver on the system. You cannot run this device driver.

Remedy: No action required

MESSAGE: SHARE already installed

Command: SHARE

Cause: This error message is displayed if you try to install SHARE more than once.

Remedy: No action required

MESSAGE: Sort: Incorrect DOS version

Command: SORT

Cause: This error message is displayed if you try to run SORT on any version of MS-DOS other than 3.1.

Remedy: Use the correct versions of SORT and MS-DOS.

MESSAGE: SORT: Insufficient disk space

Command: SORT

Cause: This error message is displayed by SORT when the disc is full.

Remedy: Remove some extraneous files from the disc, and try again.

MESSAGE: SORT: Insufficient memory

Command: SORT

Cause: This error message is displayed when there is not enough memory to run the SORT program.

Remedy: Reduce the amount of data to be SORTed.

MESSAGE: Source and target diskettes are not the same format

Command: DISKCOPY

Cause: This error message is displayed by DISKCOPY because you must have the same size and kind of disks to run DISKCOPY. Example: you cannot copy from a single-sided disc to a double-sided disc.

Remedy: Reformat the target disc to be of the same type as the source disc.

MESSAGE: SOURCE diskette bad or incompatible

Command: DISKCOMP

Cause: One of the discs is unformatted or incompatible with the drive.

Remedy: Use the properly formatted disc.

MESSAGE: Specified MS-DOS search directory bad

Command: MS-DOS

Cause: This error message is displayed by MS-DOS because the SHELL command in the CONFIG.SYS file is incorrect. The place that you have told MS-DOS to find COMMAND.COM does not exist, or COMMAND.COM is not in that place.

Remedy: Ensure that the SHELL program is started from the Root directory.

MESSAGE: Strike a key when ready...

Command: MS-DOS

Cause: This prompt occurs during command processing and is always accompanied by another message. This message is also displayed if you have inserted a Pause command in a batch file. Usually, you are asked to insert disks into appropriate drives before this prompt.

Remedy: Perform the specified task, then press any alphanumeric key.

MESSAGE: Syntax error

Command: FIND

Cause: You have typed a command line which FIND cannot interpret.

Remedy: You should check to make sure that you have typed the command correctly.

MESSAGE: System transferred

Command: FORMAT, SYS

Cause: This message is displayed by FORMAT or SYS when the system files MSDOS.SYS and IO.SYS have been transferred during FORMAT or SYS command processing.

Remedy: No action required

MESSAGE: Target diskette is write protected
Correct, then strike any key

Command: DISKCOPY

Cause: This error message is displayed when the target diskette has a write protect tab covering the write-enable notch.

Remedy: Remove the tab and reinsert the diskette.

MESSAGE: Target Diskette may be unusable

Command: DISKCOPY

Cause: This error message is displayed because of read, write, or verify errors in the copying process. The target disc may be incomplete.

Remedy: Compare the two discs with DISKCOMP. Try again. A new destination disc may be necessary.

MESSAGE: Terminate batch job (Y/N)?

Command: MS-DOS

Cause: This error message is displayed by MS-DOS if you press `CTRL` C while in batch mode, MS-DOS asks you whether or not you wish to end batch processing.

Remedy: Press Y to end processing. Press N to continue the batch job.

MESSAGE: Too many files open

Command: EDLIN

Cause: This error message is displayed by EDLIN when MS-DOS could not open the file to edit on the .BAK file due to lack of system file handles.

Remedy: Increase the value of the FILES command in the CONFIG.SYS file.

MESSAGE: Track 0 bad - disk unusable

Command: FORMAT

Cause: The FORMAT command can accommodate for defective sectors on the disc except for those near the beginning.

Remedy: Try formatting another disc.

MESSAGE: Unable to create a directory

Command: MS-DOS, MKDIR

Cause: This error message is displayed when MS-DOS could not create the directory you specified.

Remedy: Check to see that there is not a name conflict (you may have a file by the same name) or the disc may be full.

MESSAGE: Unrecognized command in CONFIG.SYS

Command: CONFIG.SYS

Cause: This error message is displayed by MS-DOS when there is an invalid command in your CONFIG.SYS file.

Remedy: Refer to the chapter titled *System Configuration* for a list of valid commands.

MESSAGE: Unrecoverable error in Directory
Convert directory to file (Y/N)?

Command: CHKDSK

Cause: This error message is displayed by CHKDSK when there has been an unrecoverable error in directory.

Remedy: If you respond Y to this prompt, CHKDSK will convert the bad directory into a file. You can then fix the directory yourself or delete it.

MESSAGE: Unrecoverable Read Error on Drive x:
Side y, Track z

Command: DISKCOMP, DISKCOPY

Cause: This error message is displayed after several

unsuccessful attempts are made to read the data from the specified track and sector on the disc in the indicated drive.

Remedy: The disc is bad. RECOVER as many files as possible, copy the files to another disc, and try reformatting.

MESSAGE: Unrecoverable Write Error on Drive x:
Side y, Track z

Command: DISKCOPY

Cause: This error message is displayed after several unsuccessful attempts are made to write data to the specified track and sector on the disc in the indicated drive.

Remedy: The disc is bad. RECOVER as many files as possible, copy the files to another disc, and try reformatting.

MESSAGE: VDISK not installed - buffer too small

Command: VDISK

Cause: This error message is displayed when the virtual disc drive cannot be installed due to an incorrect buffer size.

Remedy: Change buffer size if necessary.

MESSAGE: VDISK not installed - insufficient
memory

- Command:** VDISK
- Cause:** This error message is displayed when the virtual disc drive cannot be installed due to an insufficient memory. If less than 64K of system memory would be left after the virtual disc is installed, this message will be displayed.
- Remedy:** Change buffer size if necessary.
- MESSAGE:** VDISK not installed - no extended memory
- Command:** VDISK
- Cause:** This error message is displayed when the /E switch was specified, but the system does not have extended memory, or the amount of available extended memory is insufficient to contain the virtual disc even after adjusting the parameters.
- Remedy:** Ensure that you have enough extended memory to run VDISK, or reduce the size of the VDISK.
- MESSAGE:** Volume label (11 characters, ENTER for none)?
- Command:** FORMAT
- Cause:** This message is displayed by FORMAT when you specify the /V switch in the FORMAT command.
- Remedy:** Specify a volume label or press the

Enter key to indicate that you do not want a volume label for the disc.

MESSAGE: WARNING, ALL DATA ON NON-REMOVABLE DISK DRIVE X WILL BE LOST, Proceed with Format (Y/N)?

Command: FORMAT

Cause: This error message is displayed by FORMAT when there is data on the hard disc that you are trying to format.

Remedy: If you want to lose the data and format the disc, press Y (for "Yes"). If you do not want the files on your hard disc erased press N (for "No"). Copy the files to a floppy disc and repeat the FORMAT command.

MESSAGE: Warning - directory full

Command: RECOVER

Cause: This error message is displayed by RECOVER when the root directory is too full for RECOVER processing.

Remedy: Delete some files in the root directory to free space.

MESSAGE: Warning: Read error in EXE file

Command: EXE2BIN

Cause: This error message is displayed by EXE2BIN when the amount read was less than the size of the header. This is a warning message only.

MESSAGE: (xxxx) of (xxxx) Bytes recovered

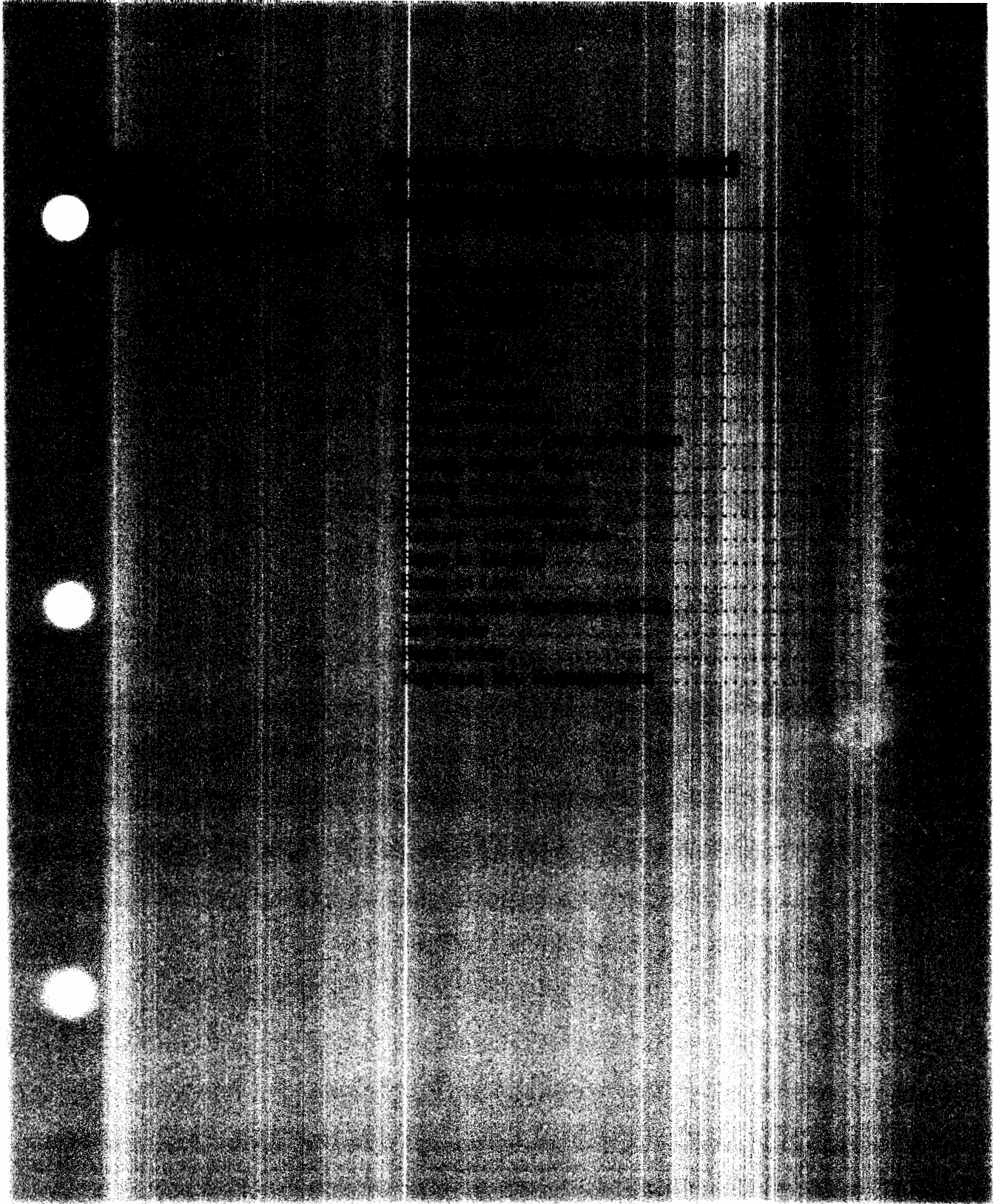
Command: RECOVER

Cause: This error message displayed by RECOVER tells you how many bytes MS-DOS was able to recover of the disc or file.

MESSAGE: 10 Mismatches - ending compare

Command: COMP

Cause: This informational message is displayed by COMP when 10 mismatches are found. The compare operation is terminated.



B

Extended Screen and Keyboard Control

This chapter explains the use and features of ANSI.SYS. This device driver provides support for ANSI Standard Terminal Escape sequences. In order to utilize these features, the extended screen and keyboard control device driver ANSI.SYS must be installed. To install this driver, insert the command line

```
DEVICE=[<d>:][<path>]ANSI.SYS
```

in the CONFIG.SYS file. Additional information regarding installation of device drivers is contained in the chapter titled *System Configuration*.

Extended screen and keyboard control codes are valid for programs using Standard Input, Standard Output, and Standard Error Message devices. These devices may be accessed through MS-DOS system calls 01H, 02H, 06H, 07H, 09H, 0AH, and 40H.

Control Sequence Syntax

All control sequences use the syntax shown below:

ESC [<parameters> <command identifier>

ESC	This is the one byte ASCII Escape code. All control sequences begin with this character. ESC is 1BH or 27 decimal.
[This is the second character in the sequence.
<parameters>	These are optional parameters. Parameters can consist of either alpha character strings or numeric values. Numeric values will be indicated with the '#' character. If a parameter is omitted, or a value of 0 is entered, the default value of the parameter is assumed.
<command identifier>	This is a single alphabetic character which identifies the control sequence.

Control Sequences

The control sequences recognized by ANSI.SYS are listed below. The screen control sequences are listed first, followed by the keyboard sequences.



Cursor Position **Operation**

This sequence moves the current cursor position to the specified row and column. The row is specified by the first number; the column by the second. If the row and/or column number is omitted, the default value of 1 is substituted. If both numbers are omitted, the cursor is moved to the HOME position (Row 1, Column 1).

Syntax

```
ESC[#: #H
```

Example

The following sequence moves the cursor to Row 10, Column 5.

```
ESC[10;5H
```

Cursor Up **Operation**

This sequence moves the cursor up a specified number of rows. If the cursor is already in the top row, this sequence is ignored by ANSI.SYS. The default number of rows to move is 1.

Syntax

```
ESC[#A
```

Example

The following sequence moves the cursor up 5 rows.

```
ESC[5A
```

Cursor Down **Operation**

This sequence moves the cursor down a specified number of rows. If the cursor is already on the bottom row, this sequence is ignored by ANSI.SYS. The default number of rows to move is 1.

Example

```
ESC[#B
```

Example

The following sequence moves the cursor down 10 rows.

```
ESC[10B
```

Cursor Forward

Operation

This sequence moves the cursor forward (to the right) a specified number of columns. If the cursor is in the rightmost column, this sequence is ignored by ANSI.SYS. The default number of columns to move is 1.

Syntax

```
ESC[#C
```

Example

The following sequence moves the cursor forward 8 columns.

```
ESC[8C
```


Cursor Backward **Operation**

This sequence moves the cursor backward (to the left) a specified number of columns. If the cursor is in the leftmost column, this sequence is ignored by ANSI.SYS. The default number of columns to move is 1.

Syntax

```
ESC[#D
```

Example

The following sequence moves the cursor backward 2 columns.

```
ESC[2D
```

Horizontal and Vertical Position

Operation

This sequence moves the current cursor position to the specified row and column. The row is specified by the first number; the column by the second. If the row and/or column number is omitted, the default value of 1 is substituted. If both numbers are omitted, the cursor is moved to the HOME position (Row 1, Column 1).

Syntax

```
ESC[r;cf
```

Example

The following sequence moves the cursor to Row 10, Column 5.

```
ESC[10;5f
```



Cursor Position Report

Operation

This sequence is returned in response to an inquiry. The sequence is returned through the Standard Input device. The first parameter specifies the current row and the second parameter specifies the current column.

Syntax

```
ESC I # ; # R
```

Example

The following sequence reports the current cursor row as 10 and column position as 5.

```
ESC I 10 ; 5 R
```

Device Status Report **Operation**

This sequence requests a Cursor Position Report from the console driver. The Report is returned in the format shown for the Cursor Position Report above.

Syntax

```
ESC[Ln
```

Example

The following sequence requests the current device status.

```
ESC[Ln
```

After issuing this sequence, a Cursor Position Report may be read from the Standard Input.

Save Cursor Position

Operation

This sequence saves the current cursor position so that it can be recalled later by the Restore Cursor Position sequence.

Syntax

```
ESCIs
```

Example

The following sequence saves the current cursor position.

```
ESCIs
```

Restore Cursor Position

Operation

This sequence restores the cursor position to the location it had when the console driver last received the Save Cursor Position sequence.

Syntax

```
ESCl
```

Example

The following sequence restores the cursor position:

```
ESCl
```

Erase in Display **Operation**

This sequence clears the entire screen and homes the cursor.

Syntax

```
ESC[2J
```

Example

The following sequence clears the screen.

```
ESC[2J
```

Erase in Line **Operation**

This sequence erases from the character under the cursor to the end of the row.

Syntax

ESC[K

Example

The following sequence erases to the end of the current row.

ESC[K

Set Graphics Rendition (SGR)

Operation

This sequence will set the display attributes for the character at the current cursor position. The attributes will apply to all characters following the current cursor position until the next SGR is encountered. Each character may have more than one display attribute. The following table lists the numbers assigned to each of the possible characteristics.

Parameter	Graphics Characteristic
0	All attributes off (White foreground, Black background)
1	Bold On (High Intensity)
4	Underscore On (monochrome display adapter only)
5	Blink On
7	Reverse Video
8	Cancel On (invisible)
30	Black foreground
31	Red foreground
32	Green foreground
33	Yellow foreground
34	Blue foreground
35	Magenta foreground

36	Cyan foreground
37	White foreground
40	Black background
41	Red background
42	Green background
43	Yellow background
44	Blue background
45	Magenta background
46	Cyan background
47	White background

Syntax

```
ESC[i;...;m
```

Example

The following sequence sets the SGR at the current cursor position to Green foreground on a Black background.

```
ESC[32;40m
```

Set Mode **Operation**

Invokes the screen width or type specified by the parameter.

Parameter	Screen Width or Type
0	40×25 Black & White
1	40×25 Color or Gray Scale
2	80×25 Black & White
3	80×25 Color or Gray Scale
4	320×200 Color or Gray Scale
5	320×200 Black & White
6	640×200 Black & White
7	Wrap at end of line. Typing past end-of-line results in new line.)

Syntax

ESC [=#h

or

ESC [=h

or

ESC [=h

or

ESC [?h

Example

The following sequence sets the screen to the 80×25 Black & White mode.

```
ESC I =2h
```



Reset Mode **Operation**

This function is identical to Set Mode except that parameter 7 will reset wrap at end-of-line mode. (Characters past end-of-line are thrown away.)

Example

```
ESC[#1
```

or

```
ESC[=1
```

or

```
ESC[=01
```

or

```
ESC[?71
```

Keyboard Key Reassignment

Operation

The first ASCII code in the control sequence defines which code is being mapped. The remaining numbers define the sequence of ASCII codes generated when this key is intercepted. Note that there is one exception; if the first code in the sequence is zero then the first and second code make up an extended ASCII definition.

Syntax

```
ESC[#;#;...#p
```

or

```
ESC["string"p
```

or

```
ESC[#;"string";#;#;"string";#p
```

or any other combination of strings and decimal numbers.

Example

1. Reassign the Q and q keys to the A and a keys (and vice versa):

```
ESC[65;81p      A becomes Q
```

```
ESC[97;113p     a becomes q
```

```
ESC[81;65p      Q becomes A
```

```
ESC[113;97p     q becomes a
```

2. Reassign the `F10` key to a `DIR` command followed by a carriage return:

```
ESC[O;68;"DIR";13p
```

The `O;68` is the extended ASCII code for the `F10` key; `13` decimal is a carriage return.

C

Non-US Keyboard Layouts

France/French	C-2
Germany/German	C-3
Italy/Italian	C-4
Spain/Spanish	C-5
U.K./English	C-6

C

Non-US Keyboard Layouts

Keyboard Layouts

Layouts for the following non-U.S. keyboards are on the following pages:

France/French

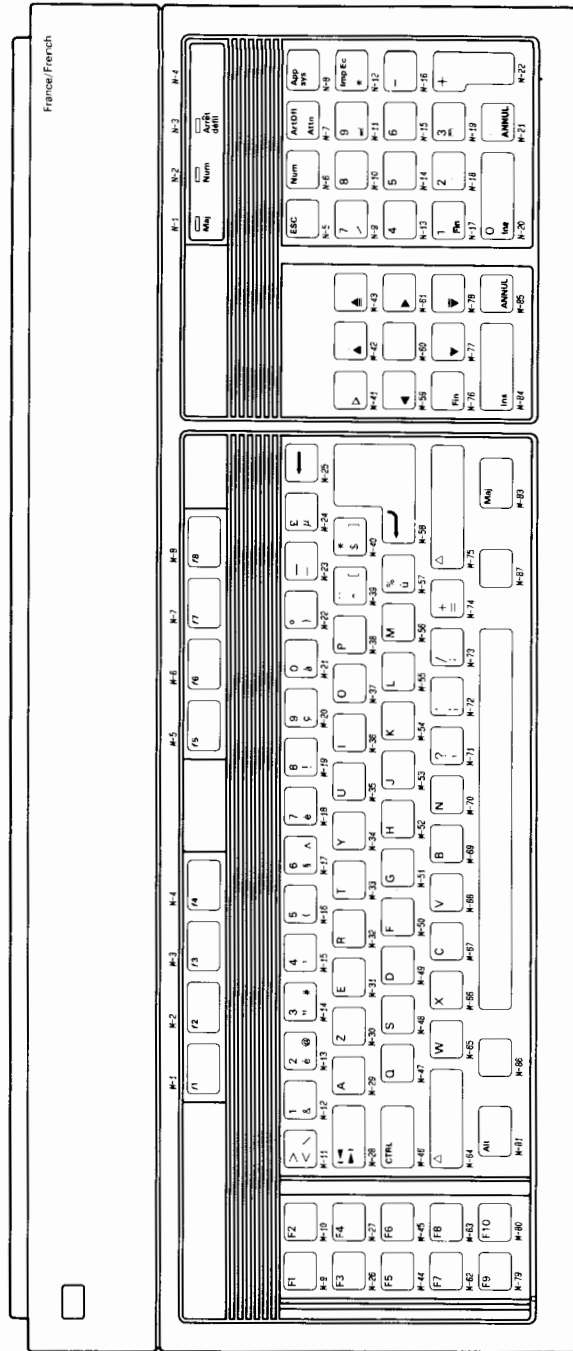
Germany/German

Italy/Italian

Spain/Spanish

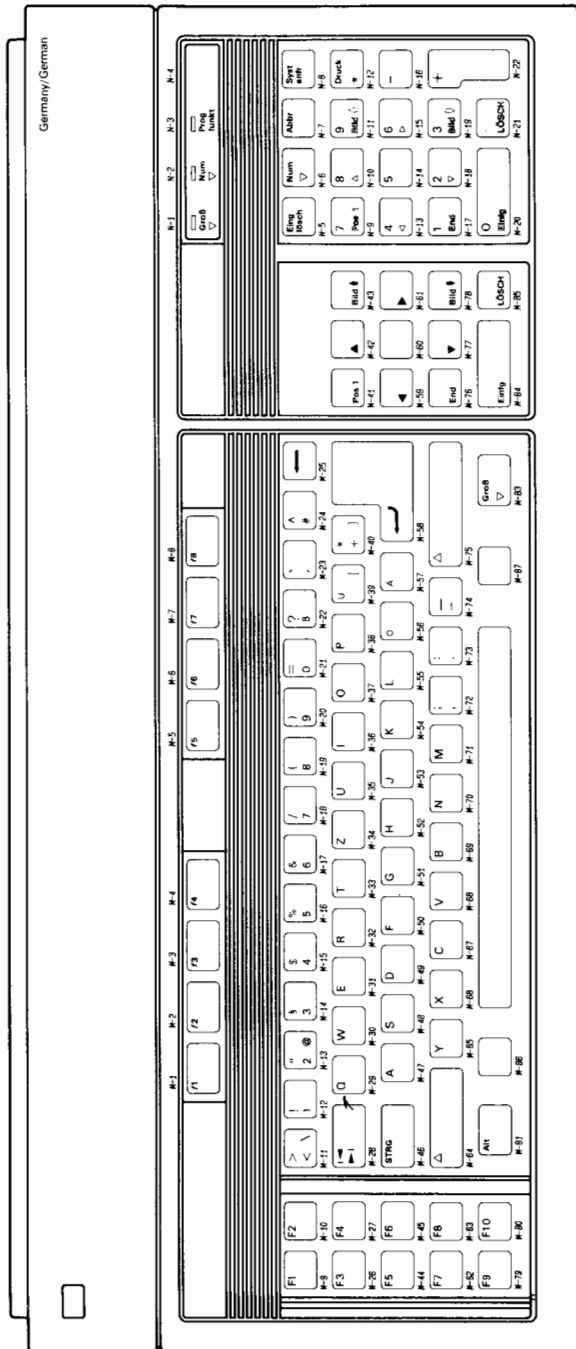
U.K./English

France/French



C-2 Non-US Keyboard Layouts

Germany/German



Non-US Keyboard Layouts C-3

D

**Disc and Disc Drive
Compatibility**

D

Disc and Disc Drive Compatibility

This appendix describes the level of compatibility between 1.2 MByte discs and 360 KByte discs.

Despite the outward similarity of the 360 KByte discs and 1.2 MByte discs, the two technologies are not completely compatible. The following table shows which combinations of disc drive types, disc media types, and formatting options may be used reliably.

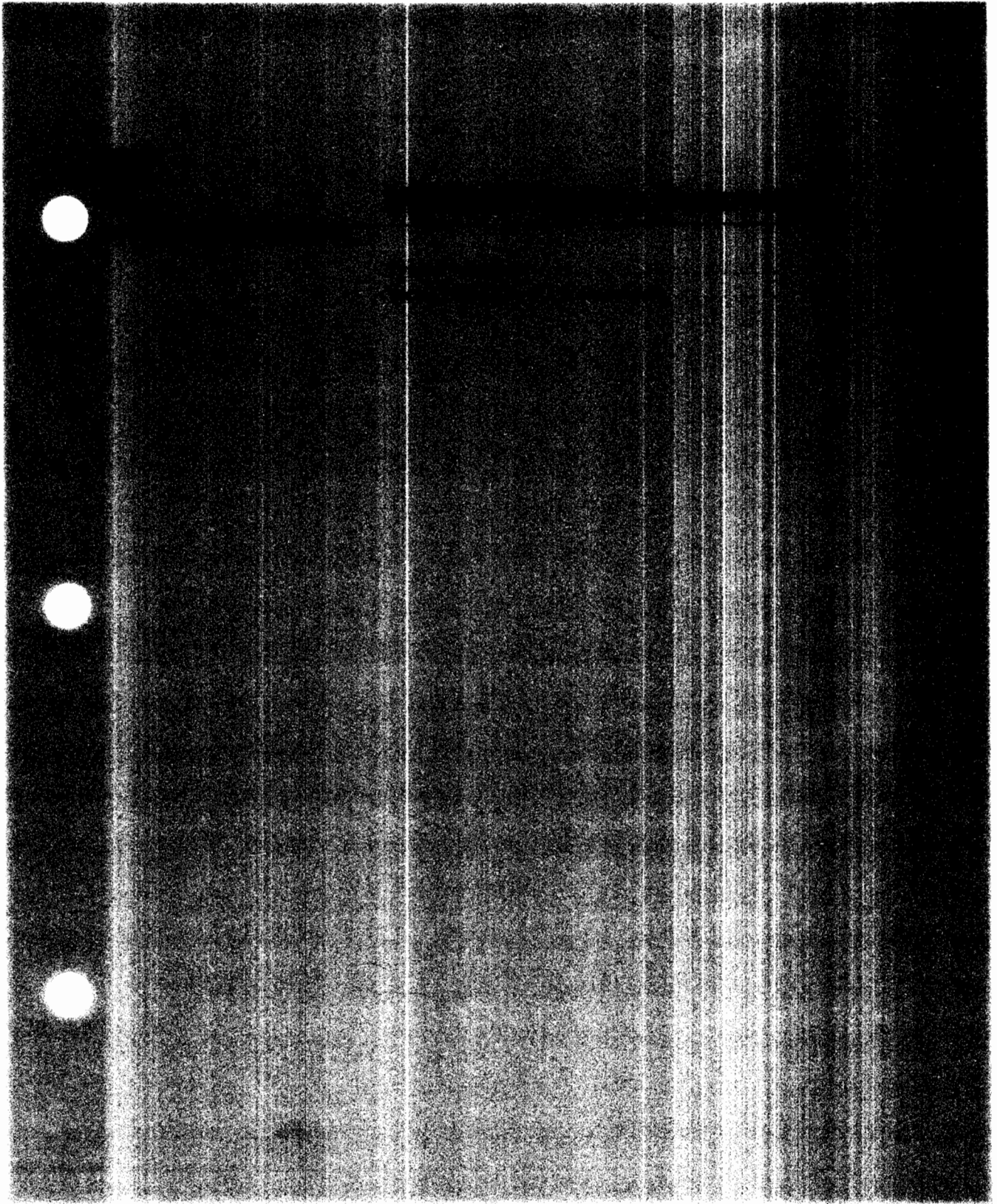
Media and Drive Combination:		Disc Media Was Formatted In:		
Drive Type	Media	360 Kb drive	1.2 Mb drive with /4 option	1.2 Mb drive
360Kb	1.2Mb	No	No	No
1.2Mb	1.2Mb	No	No	Read/Write
1.2Mb	360Kb	Read/Write*	Read/Write*	No
360Kb	360Kb	*Read/Write	No	No

* If you write on 360Kb media with a 1.2Mb drive, then no matter how the disc was formatted, the disc will be useable (reliably) only in a 1.2Mb drive from that time on. The disc must be reformatted to be used reliably in a 360Kb drive.

Caution



Combinations that are marked “No” in the table above may or may not function without data loss. These combinations are inherently unreliable, and cannot be used without an extremely high risk to the integrity of the data being transferred between the disc and system memory. Hewlett-Packard recommends that you **never** use a combination which is marked “**No**” in the table above, and avoid using disc media in non-matching drive types wherever possible.



E

MS-DOS Syntax Summary

This appendix lists a brief definition and the syntax for all Vectra MS-DOS 3.1 commands.



Syntax Notation

The following symbols are used when describing command syntax in this appendix:

- CAPS** Words in capital letters indicate commands or portions of commands that must be typed exactly as shown.
- < >** Words enclosed by angle brackets represent data you must enter. When the angle brackets enclose lower-case text (for example, <path>), you must supply the entry defined by the text.
- []** Words enclosed in square brackets are optional.
- |** A vertical bar between two command options means one or the other must be used. When used with an MS-DOS filter, a vertical bar indicates a pipe.
- /** A front slash designates a switch.
- ...** An ellipsis indicates that an entry may be repeated as many times as needed or desired.

All other punctuation, such as commas, colons, slash marks (/), and equal signs, must be entered exactly as shown.

Vectra MS-DOS

3.1 Commands

ASSIGN

The ASSIGN command substitutes one disc drive designator for another.

```
[<mdir>]ASSIGN [<x>[=]<y>[...]]
```

ATTRIB

The ATTRIB (attribute) command sets or clears the Read-Only attribute flag for one or more files.

```
[<mdir>]ATTRIB [+R|-R] [<dir>]<file name>
```

BACKUP

The BACKUP command makes back up copies of files onto flexible discs.

```
[<mdir>]BACKUP <d>:[<path>][<file name>] <d>:  
[ /S ] [ /M ] [ /A ]  
[ /D : <mm-dd-yy> ]
```

BREAK

Sets **CTRL** **Break** and **CTRL** **C** checking to include disc reads and writes.

```
BREAK [ON|OFF]
```

CHDIR

The CHDIR (Change Directory) command changes the current directory on the specified drive.

```
CHDIR [<d>:] <path>
```

CHKDSK

The CHKDSK (Check Disc) command tests the integrity of a disc and reports the capacity and usage of that disc and system memory.

```
[<sdir>]CHKDSK [<dir>][<file name>] [/F][/V]
```

CLS

The CLS (Clear Screen) command clears the display screen.

```
CLS
```

COMMAND

COMMAND starts COMMAND.COM, the MS-DOS Command Processor.

```
[<sdir>] COMMAND [<sdir>] [/P]  
                [/C<command string>]
```

COMP

The COMP command compares two files and reports the differences.

```
[<sdir>] COMP [<dir>] [<file name1>] [<dir>]  
                [<file name2>]
```


COPY

The COPY command duplicates one or more files to the same disc or directory, or to another disc or directory.

```
COPY [<dir>]<sfile>[/A][/B] [<dir>][<dfile>]  
    [/A][/B][/V]
```

or

```
COPY [<dir>]<sfile1>[/A][/B]  
    [+ [<dir>]<sfile2>[/A][/B]...]  
    [<dir>]<dfile>[/A][/B]
```

CTTY

CTTY allows you to substitute a character I/O device for the standard console (keyboard and screen).

```
CTTY <device name>
```

DATE

The DATE command displays or sets the current system date.

```
DATE [<mm>-<dd>-<yy>] or  
    [<mm>/<dd>/<yy>]
```

DEL

The DEL (Delete) command removes one or more files from a disc. (Same as ERASE.)

```
DEL [<dir>]<file name>
```

DIR

The DIR (Show Directory) command displays the names of files in a directory.

```
DIR [<dir>][<file name>][/P][/W]
```



DISKCOMP

DISKCOMP makes a track-by-track comparison of two compatible flexible discs.

```
[<sdir>]DISKCOMP[<d>:[<d>:] ] [/L] [/B]
```

DISKCOPY

The DISKCOPY command makes an exact copy of the contents of one flexible disc to another flexible disc.

```
[<sdir>]DISKCOPY[<d>:[<d>:] ] [/L]
```

ERASE

The ERASE command removes one or more files from a disc. (Same as DEL.)

```
ERASE [<dir>]<file name>
```

EXE2BIN

The EXE2BIN (EXEcutable to BINary) command converts files from .EXE (executable) format to .COM (command) or .BIN (binary) format.

```
[<sdir>] EXE2BIN [<dir1>] <sfile>  
                [<dir2>][<dfile>]
```

EXIT

The EXIT command exits the command processor and returns to a previous level, such as PAM, if one exists.

```
EXIT
```

FC

The FC (File Compare) command compares the contents of two files and generates a report.

```
[<mdir>] FC [/A] [/B] [/C] [/L] [/LB <n>]
          [/N] [/T] [/W] [/<nnnn>]
          [<dir>]<file name> [<dir>]<file name2>
```

FDISK

FDISK creates a variable-sized DOS partition on a hard disc.

```
FDISK
```

FIND

FIND searches for a specified text string in one or more input files.

```
[<mdir>]FIND [/V][/C][/N] <"text string">
          [[<dir>][<file name>]...]
```

FORMAT

The FORMAT command prepares a disc for use by MS-DOS.

```
[<mdir>]FORMAT [<d>:][/S][/P]
              [/V][/B][/1][/4][/B]
```

GRAFTABL

The GRAFTABL command loads the character fonts for characters to be displayed by the Multi-mode Video display adapter in the graphics mode.

```
[<mdir>]GRAFTABL
```

GRAPHICS

The GRAPHICS command outputs the contents of a graphics display to a printer.

```
[<mdir>]GRAPHICS[<printer>] [/R] [/B]
```

JOIN

The JOIN command allows a disc drive to be logically joined to a subdirectory on another drive.

```
[<mdir>]JOIN <d>: <mdir>  
[<mdir>]JOIN <d>: /D  
[<mdir>]JOIN
```

KEYBUS

The KEYBUS program replaces the ROM BIOS keyboard routines to provide support for non-U.S. keyboards.

```
[<mdir>]KEYBUS
```

LABEL

The LABEL command adds, modifies, or deletes the disc drive volume label.

```
[<mdir>]LABEL [<d>:][<volume label>]
```

MKDIR

The MKDIR (Make Directory) command creates a subdirectory on a disc.

```
MKDIR [<d>:]<path>
```

MODE

The MODE command controls the parameters for the multi-mode Video and Serial/Parallel adapter cards.

```
[<mdir>]MODE <display>  
[<mdir>]MODE [<display>],<adjust>[,T]  
[<mdir>]MODE COM#[ : ]<baud>[,<parity>  
    [,<databit>[,<stopbit>[,P]]]  
[<mdir>]MODE LPT#[ : ][[<horz>][, [<vert>]][,P]]  
[<mdir>]MODE LPT#[ : ]=COM#
```

MORE

MORE reads data from the standard input device and prints it to the standard output device one screen at a time.

```
[<mdir>]MORE
```

PATH

The PATH command sets an alternative directory search path.

```
PATH [[<d>:]<path>[[ ; [<d>:]<path>]...]]
```

PRINT

The PRINT command prints one or more files while you continue to process other commands (background printing).

```
[<sdir>]PRINT[[<dir>][<file name>]...]
          [/D:<device>][/B:<buffsize>]
          [/U:<busyticks>][/M:<maxticks>]
          [/S:<timeslice>]
          [/Q:<queuesize>][/C][/T][/P]
```

PROMPT

The PROMPT command changes the system prompt.

```
PROMPT [<text>]
```

RECOVER

The RECOVER command recovers data files, directories, or entire discs with defective sectors.

```
[<sdir>]RECOVER [<dir>]<filename>
[<sdir>]RECOVER <d>:
```

RENAME

The RENAME command changes the name of a file.

```
RENAME [<dir>]<sfile><dfile>
```

RESTORE

The RESTORE command restores files to disc after they have been copied using BACKUP.

```
[<mdir>]RESTORE <d>: [<dir>]
    [<file name>] [/S][/P]
```

RMDIR

The RMDIR (Remove Directory) command deletes a subdirectory from a disc.

```
RMDIR [<d>:]<path>
```

SET

The SET command displays or sets the current MS-DOS environment.

```
SET [<label>]=[<parameter>]]
```

SHARE

The SHARE command provides support for file sharing in a networking environment.

```
[<mdir>]SHARE [/F:<filesize>] [/L:<locks>]
```

SORT

The SORT command reads data from the standard input device, sorts it, and prints it to the standard output device.

```
[<mdir>]SORT [/R][/+<n>]
```

SUBST

The SUBST (Substitute) command substitutes a drive designator for a path.

```
[<mdir>]SUBST <d>: <d>:<path>  
[<mdir>]SUBST <d>: /D  
[<mdir>]SUBST
```



SYS

The SYS (System) command transfers the MS-DOS system files to the specified disc.

```
[<mdir>]SYS <d>: [/P]
```

TIME

The TIME command displays or sets the system time.

```
TIME [<hh:mm>[<:ss>[<.xx>]]]
```

TREE

The TREE command displays the directory paths on your disc.

```
[<mdir>]TREE [<d>:] [/F]
```

TYPE

The TYPE command displays the contents of a file to the screen.

```
TYPE [<dir>]<file name>
```


VER

The VER (Version) command displays the version number of your MS-DOS operating system.

```
VER
```

VERIFY

The VERIFY command sets the verify switch on or off.

```
VERIFY [ON|OFF]
```

VOL

The VOL (Volume) command displays your disc volume label.

```
VOL [<d>:]
```

Index

A Active Drive, 2-17, 2-19, 3-2
ANSI.SYS, 7-15, 7-17, 12-92, B-1
Applications, 1-1, 2-2, 7-4
Archive file attribute, 2-13, 12-101
ASCII files, 12-31
ASSIGN, 12-6, 12-46, 12-70, 12-106
ATTRIB, 2-13, 12-8
AUTOEXEC Batch file, 1-4, 4-5
AUX:, 2-8, 8-1, 12-32

B BACKUP, 2-13, 4-17, 7-18, 12-10, 12-31, 12-96, 12-106
BASIC, 2-7
Batch Processing,
 AUTOEXEC.BAT, 1-4, 4-5
 Batch Command File, 3-2, 4-2
 Program Exit Codes, 4-16
 Replaceable parameters, 4-6, 4-21,
Batch Commands,
 ECHO, 4-10
 FOR, 4-12
 GOTO, 4-14
 IF, 4-16
 PAUSE, 4-18
 REM, 4-20
 SHIFT, 4-8, 4-21
BREAK, 4-5, 7-3, 12-13
BUFFERS, 7-4

C Character fonts, 12-67
CHDIR, 2-15, 2-19, 12-17, 12-76, 12-98
CHKDSK, 2-17, 4-3, 12-17, 12-93, 12-118
CLOCK\$, 2-9
CLS, 12-21
COMP, 12-24, 12-44
CON:, 2-8, 4-2, 8-1
Configuration Commands,
 BREAK, 7-3
 BUFFERS, 7-4
 COUNTRY, 7-6
 DEVICE, 7-8
 FCBS, 7-9
 FILES, 7-11
 LASTDRIVE, 7-13
 SHELL, 7-14
CONFIG.SYS file, 7-1, 12-16, 12-32, 12-35, 12-40, 12-62,
 12-109, 12-111
COM1:, 2-8, 12-80
COM2:, 2-8, 12-80
COMMAND.COM, 3-1, 7-14, 12-22, 12-51, 12-62, 12-102,
 12-109
COMMAND
 Basic Information, 3-1, 12-1
 External, 3-1
Command Line, 3-1, 3-3, 12-3
Command Line Template, 5-1
COPY, 4-2, 7-18, 12-12, 12-27, 12-96
COUNTRY, 7-6, 12-37, 12-42, 12-72
CTTY, 8-1, 12-32
Current Directory, 2-17, 2-19, 2-21, 3-2, 3-4, 12-15,

D DATE, 12-34, 12-111
DEBUG Utility,
 Assemble command, 9-13
 Compare command, 9-16
 Dump command, 9-17
 Enter command, 9-19
 Error Messages, 9-46
 Fill command, 9-21
 Go command, 9-22
 Hex command, 9-25
 Input command, 9-26
 Load command, 9-27
 Move command, 9-29
 Name command, 9-30
 Output command, 9-33
 Parameters, 9-4
 Procedure command, 9-34
 Quit command, 9-35
 Register command, 9-36
 Search command, 9-39
 Trace command, 9-40
 Unassemble command, 9-42
 Write command, 9-44
DEL, 12-36
DEVICE, 7-8
Devices, 2-6, 2-8, 2-12, 8-2, 8-3, 12-32
Device Drivers, 7-1, 7-15, 8-2
DIR, 2-10, 12-38
Directories,
 Definition, 2-14
 Entries, 2-14, 2-15, 7-19
 Root, 2-14, 2-15, 2-19, 6-10, 12-15, 12-61, 12-98
 Subdirectories, 2-14, 2-15, 12-16, 12-84, 12-112
 Tree structure, 2-16, 12-112
Disc and Disc Drive Compatibility, D-1
Discs,
 Compatibility, D-1

- Cylinders, 6-1
- Formatting, 6-5, 6-10
- Hard, 2-15, 6-1
- High Capacity, 2-15, 12-63, D-1
- MS-DOS (System), 1-2
- Standard Capacity, 2-15, 12-65, D-1
- Supplemental, 1-2
- Virtual, 2-14, 12-7, 12-105
- Volume label, 6-10, 12-73, 12-119
- DISKCOMP, 12-8, 12-26, 12-42, 12-106
- DISKCOPY, 7-18, 12-12, 12-45, 12-96
- Drive designator, 2-17, 7-13, 12-3

E Editing keys, 5-1

- EDLIN, 1-5, 2-7, 4-2, 7-1
- EDLIN utility,
 - Append command, 11-11
 - Command basics, 11-4
 - Copy command, 11-13
 - Delete command, 11-17
 - Edit command, 11-19
 - End command, 11-22
 - Error messages, 11-43
 - Insert command, 11-23
 - List command, 11-26
 - Move command, 11-29
 - Page command, 11-31
 - Quit command, 11-32
 - Replace command, 11-33
 - Search command, 11-37
 - Transfer command, 11-41
 - Write command, 11-42
- End-of-file marker, 2-9, 4-2, 8-7, 12-30,
- ERASE, 12-47
- Error Messages, A-1
- EXE2BIN, 12-49
- EXIT, 12-23, 12-51

Exit Codes, 4-16
Extended memory, 7-18
Extension, 2-3, 2-11, 2-14, 3-1



F FC, 12-44, 12-52
FDISK, 6-1, 12-63, 12-110
File Allocation Table (FAT), 2-14, 2-16, 6-10, 12-61
File Control Block (FCB), 7-9
File handles, 7-11
File names,
 Definition, 2-3
 Invalid, 2-4, 2-6
 Pathnames, 2-20
 Reserved, 2-4, 2-6
 Wildcard Characters, 2-6, 2-10, 2-12, 12-29, 12-36,
File Specifier, 2-20
FILES, 7-11
Files,
 Attributes, 2-13, 2-14, 12-9
 Clusters, 2-16, 12-21, 12-23, 12-98
 Definition, 2-1
 Extensions, 2-3, 2-11, 2-14, 3-1
 Filename, 2-3, 2-11, 2-14
 Sharing, 12-105
 Time and Date stamp, 2-14, 12-13
Filters, 8-6, 12-61, 12-83, 12-103
FIND, 8-6, 12-83, 12-59, 12-104
FORMAT, 6-5, 6-10, 12-8, 12-61, 12-74, 12-93, 12-109, 12-119
Function Keys, 2-8, 4-2, 5-1

G GRAFTABL, 12-64, 12-66
GRAPHICS, 5-5, 12-64, 12-65
Graphics mode, 12-64

H Hidden file attribute, 2-14, 12-37, 12-41
HP Mouse, 1-2
HP Vectra,
 Documentation, 1-2
 Technical Reference Manual, 1-3
HP Touch Screen, 1-2

J JOIN, 12-7, 12-67, 12-110

K Keyboard Click volume, 5-7
Keyboard, Functions in MS-DOS, 5-1
Keyboard, Layouts, Non-U.S.
 France, 12-76, C-2
 Germany, 12-76, C-3
 Italy, 12-76, C-4
 Spain, 12-76, C-5
 U.K., 12-75, C-6
Keyboard, U.S. 12-71, 5-1
KEYBUS, 12-71

L LABEL, 12-8, 12-73, 12-119
LASTDRIVE, 7-13, 12-110
LINK utility,
 Command prompts, 10-23
 Error Messages, 10-36
 Examples, 10-42
 Files, 10-12

Response files, 10-20
Segments, 10-4
Switches, 10-31
LPT1:, 2-8, 12-82
LPT2:, 2-8, 12-82
LPT3:, 2-8, 12-82

M Memory Addressing, 9-7
MKDIR, 2-16, 12-75, 12-98
MODE, 12-77
MORE, 8-6, 12-83, 12-104
MS-DOS,
 Abort commands, 4-4, 4-18, 5-6, 7-3, 12-13
 Applications, 1-1
 Booting, 6-3
 Command line, 3-2
 Command Syntax, E-1
 Devices, 2-6, 8-21
 Editing keys, 5-1
 Error messages, A-1
 Environment, 4-8, 12-25, 12-90, 12-99
 Filters, 8-6, 12-61, 12-83, 12-103
 IO.SYS, 2-13, 12-62, 12-108
 Macro Assembler, 1-3
 MSDOS.SYS, 2-13, 12-62, 12-108
 Partitions, 6-2
 Programmer's Reference, 1-2, 7-15
 Syntax Notation, 12-2, E-1
 System date and time, 12-34, 12-110
 System prompt, 2-18, 3-2, 3-2, 12-90

MS-DOS Commands,

ASSIGN, 12-6, 12-14, 12-46, 12-47, 12-73, 12-106
ATTRIB, 2-13
BACKUP, 2-13, 4-17, 7-18, 12-10, 12-100, 12-106
BREAK, 4-5, 12-13
CHDIR, 2-15, 2-1, 12-15
CHKDSK, 2-18, 4-3, 12-17
CLS, 12-21
COMMAND, 12-22
COMP, 12-24
COPY, 4-2, 7-18, 12-27
CTTY, 8-1, 12-32
DATE, 12-34
DEBUG, 1-3, 1-5, 9-1
DEL, 12-36
DIR, 2-10, 12-38
DISKCOMP, 12-42
DISKCOPY, 7-18, 12-8, 12-45
EDLIN 1-5, 2-7, 4-2, 11-1
ERASE, 12-47
EXE2BIN, 12-49
EXIT, 12-51
External, 3-1
FC, 12-52
FDISK, 6-1, 12-63, 12-106
FIND, 12-59
FORMAT, 6-5, 6-10, 12-61, 12-106
GRAFTABL, 12-64
GRAPHICS, 5-5, 12-65
Internal, 3-1
JOIN, 12-8, 12-67, 12-106
KEYBUS, 12-71
LABEL, 12-8, 12-73, 12-106
LINK, 1-3, 1-5, 10-1
MKDIR, 2-16, 12-75
MODE, 12-77
MORE, 8-6, 12-83
PATH, 3-4, 12-84
PRINT, 12-8, 12-86

PROMPT, 3-2, 12-90
RECOVER, 12-93
RENAME, 12-95
RESTORE, 2-13, 4-17, 7-18, 12-96, 12-06
RMDIR, 12-39, 12-98
SET, 4-8, 12-99
SHARE, 12-101
SORT, 12-103
SUBST, 12-7, 12-70, 12-74, 12-105, 12-106
SYS, 12-65, 12-08
TIME, 12-35, 12-110
TREE, 12-112
TYPE, 12-114
VER, 12-116
VERIFY, 12-116
VOL, 12-117
MS-DOS Filters
 FIND, 8-6, 12-59
 MORE, 8-6, 12-83
 SORT, 8-6, 12-103

N Network, 12-107
 NUL, 2-9

P PAM,
 Command processor, 7-14, 12-22, 12-51
 Documentation, 1-2
 Hidden files, 2-13
 Loading, 7-14
 Setting date and time, 12-34, 12-110
 Transferring to a disc, 6-10, 12-61, 12-108
Parallel port, 12-80
Partitions, 6-2

Path, 2-19, 2-20, 3-4, 12-16, 12-100
PATH, 3-4, 12-7, 12-84, 12-88, 12-104
Pathname, 2-20
Peripherals,
 Documentation, 1-2
Piping Standard I/O, 8-5
PRINT, 12-8, 12-82, 12-86
Printing the screen, 5-4
PROMPT, 3-2, 12-90, 12-100
PRN:, 2-8, 12-87
Programs, 2-2

R Read-only file attribute, 2-13, 12-9, 12-37, 12-101
Real-Time clock, 12-34, 12-110
RECOVER, 12-93
Redirecting Standard I/O, 8-2, 12-20
RENAME, 12-95
RESTORE, 2-13, 4-17, 7-18, 12-12, 12-96, 12-106
RMDIR, 12-16, 12-37, 12-98
ROM BIOS, 1-3, 5-5, 8-7, 12-71
Root Directory, 2-14, 2-15, 2-19, 12-61, 12-75

S Serial port, 12-79
SET, 4-8, 12-83, 12-92, 12-99
SHARE, 7-9, 12-12, 12-101
SORT, 8-6, 12-83
Standard Auxiliary Device, 7-3
Standard Error Device, 8-8
Standard Input/Output, 5-5, 7-3, 8-1, 12-32, 12-87, 12-105
Standard Printer Device, 7-3
Subdirectories, 2-14, 2-15, 12-75, 12-98

SUBST, 7-13, 12-8, 12-70, 12-84, 12-106
SYS, 12-65, 12-113
System Configuration commands,
 BREAK, 7-3
 BUFFERS, 7-4
 COUNTRY, 7-6, 12-37, 12-42, 12-115
 DEVICE, 7-8
 FCBS, 7-9
 FILES, 7-11
 LASTDRIVE, 7-13
 SHELL, 7-14
System file attribute, 2-13
System prompt, 2-18, 3-2, 3-3
System date and time, 12-34, 12-110

T TIME, 12-110
TREE, 12-112
TYPE, 12-114

V VDISK.SYS, 2-14, 7-8, 7-15, 7-17, 12-7
VER, 12-116
VERIFY, 12-31, 12-117
Video display, 12-79
VOL, 12-74, 12-123
Volume label, 6-10, 12-62, 12-73, 12-119

W Wildcard Characters, 2-6, 2-10, 2-12, 12-29, 12-31

