Microsoft® MS-DOS® 3.3 Reference

HP 9000 Series 300/800 Computers

HP Part Number 98870-90050

Hewlett-Packard Company
3404 East Harmony Road, Fort Collins, Colorado 80525

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 MANUAL, 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 direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

WARRANTY

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

October 1988...Edition 1

September 1989...Edition 2

Contents

1

Using Reference Information

This manual contains information about MS-DOS (DOS) commands, utilities, and facilities. The table shows how the manual can help you.

What You Want To Do	Where To Get Information
You want to use a DOS command	Chapters 1 and 2 (and Appendix C) provide information about applicability, purpose, syntax, examples, and use. Appendix A explains error recovery.
You want to customize your system.	Chapter 3 and Appendix B discuss commands related to these tasks. Using this information, and the information in your user's guide, you can tailor a system to your specific needs.
As an expert, you want to use batch and object files	Chapters 3 through 6 provide information about these areas.

To provide information about using the manual, this chapter:

- Describes the contents of the manual by chapters.
- Explains the syntax and symbols used to document commands.
- Discusses internal and external MS-DOS commands.
- Defines often used terms.
- Discusses compatibility among versions of MS-DOS.

Contents of Subsequent Chapters and Appendices

Except for reading Chapter 1 first, you can read chapters according to your needs.

Ch/Ap	Name	Description
2	MS-DOS 3.3 Commands	Contains an alphabetical list of all DOS commands.
3	Batch File Commands	Describes how to use a batch file and explains the subcommands available in BATCH.
4	EDLIN Commands	Describes how to use EDLIN (a line editor) and explains how to use its commands.
5	DEBUG Commands	Describes how to use the facility called DEBUG, which helps you debug binary and executable object files.
6	MS-LINK	Describes how to use the utility called MS-LINK, which links modules of 80286 object code to one load module (an .EXE file).
A	DOS Messages	Lists and describes the error messages commonly displayed by DOS commands.
B	Configuration Commands	Contains information about the commands used to configure DOS.

Running DOS Commands

The reference material for DOS assumes you understand certain terms. This section describes commonly used terminology, DOS prompts, command syntax, error messages, and so on.

The DOS Prompt and Cursor

The DOS interface (prompt) is: a letter designating the current drive; an open angle bracket (>); and a blinking cursor (underscore). For drive C, you will probably see:

`C:\>_` or perhaps just `C>_`

While the DOS interface is crude, it does give you some information.

- The letter indicates the current disk drive (the drive having the disk on which DOS looks for files unless you specify otherwise). The letters used to label disk drives include:

  A The main flexible disk drive. Used primarily for initially starting up DOS, copying applications to a hard disk, or holding disks having data or libraries. After installing and configuring DOS, most people leave drive A empty and have DOS start up from drive C.

  B A secondary flexible disk drive typically used for a data disk. (Your system may not have this disk.)

  C The hard disk having the primary (or only) partition. Used for myriad purposes. It typically contains a root directory and the subdirectories used by DOS, applications, and yourself.

  D to Z On a hard disk, the extended partition (if it exists) can contain up to 23 logical drives, which are labeled from D through Z. Many people set up drive D, and perhaps E.

- The > implies that a command is echoed on the line as you type.
- The cursor (the underscore or box) shows the location where typed letters are inserted (displayed).

The important thing about the prompt is: you cannot run a DOS command from the keyboard unless you see the prompt somewhere on the display with the cursor sitting next to it.

The Parts of a DOS Command

To run (execute, issue, or enter) a DOS command, you always type the name of the command; and you may specify additional parts such as drives, paths, file names, extensions, parameters, and options (switches). In addition, a command can require a specific use of punctuation. Accommodating this assumes considerable knowledge on your part. The next several subsections discuss the parts of a DOS command.

Command Name

You always use a command name. The names always appear in all capital letters in the documentation, but you can type them in any combination of upper- and lower-case letters. Most people type all parts of a DOS command, including the command name, in lower-case letters.

Drives and Disks

A disk drive is a device that contains a disk that has the media on which you store directories and files. The drives used in a DOS system can have different disks. It gets redundant to keep specifying micro-flexible disk, flexible disk, hard disk, partition on a disk, and so on unless the context warrants making a distinction. Therefore, the reference manual talks about "disks" and makes distinctions when necessary.

The various drives used in a DOS system can have capacities varying from 360 Kbytes to many Mbytes. The reference material assumes you know the capacities of your drives and disks. The material also assumes you match the capacities of your drives and disks, and it assumes you know when you can mismatch them.

Whatever the type of drive and disk, you specify the drive (and therefore the disk) by entering a letter and colon (for example, *A:* specifies drive A and the disk it contains). Consequently, the reference material does not repeatedly say: "The disk in the drive that.... " The reference material simply says: "Drive A", and so on.

Devices and Device Drivers

The term has several connotations. Devices can be physical hardware such as a printer; and in some cases, a device can be an interface card such as an asynchronous communications card.

A device will have a device driver; a special type of software that facilitates communication between the device and the computer system.

Devices can have names such as *PRN, CON:*, and *LPT1* (the colon can be optional or required). Some commands require specification of the name of a device or device driver.

Paths

A path is a series of directories connected by backslashes that leads to a file. An initial backslash indicates the root directory. A backslash connects the last directory in a path with a file when you specify a path and file name. A path, including all backslashes, can contain 63 characters. A path cannot have any spaces. You can use two types of paths.

Absolute paths. start with a backslash, moving from the root directory to a file. Consider the following example:

\util\tools\local\wrench	The *\util\tools\local* is the absolute path to the file named *wrench* on the disk in the current drive.

Relative paths. do not start with a backslash, moving to a file from the current directory. Consider the following example:

tools\local\wrenches	The *tools\local* is a relative path to the file named *wrenches* from the current directory, which is *util* in the above example that shows an absolute path..

Directories

Directories contain files. Directories have names, but do not have extensions. A directory name contains one to eight characters (for example, \, *tools*, *ltrs*, *reports*, *my_123*, and so on). A directory name can be any combination of upper- and lower-case letters because DOS does not make distinctions between them.

DOS has a hierarchial file system, and this has the following implications:

- The root (highest) directory contains subdirectories which themselves can contain subdirectories, and so on. Thus, the hierarchial system looks like the root system of a tree. An initial backslash in a path always denotes the root directory. Subdirectories in a path have names other than \.
- The current directory is the directory DOS uses when you run a command and do not specify a directory. Each drive in your system has a current directory. Any directory in the file system on a disk can be the current directory for that disk.
- A parent directory is any directory that contains subdirectories. Consider the following example:

```
->                LANGS
          ------------------
      BASIC               LISP
```

  In the example, *LANGS* is the parent of *BASIC* and *LISP*. (The root directory is the parent of all directories in a file system.)

- A subdirectory is a directory that appears under a directory In the example for the above item, *BASIC* and *LISP* are subdirectories under *LANGS*.

Files

Files contain alphanumeric and special characters, or they contain images of bits, bytes, and code. You can have several types of files: text, program, object, graphics, and so on. Files have names and can have an extension. For example, in:

my_junk.txt	*my_junk* is the file name and *txt* is the extension.

Periods separate file names from extensions. A file name can have one to eight characters. An extension can have one to three characters. A name can have any combination of upper- and lower-case letters because DOS changes lower-case to upper-case.

Reserved Characters. Besides not using \, ", <, and > for obvious reasons, the table shows the characters you **cannot** use in a file name or extension.

.	]	[	?	|
/	+	*	:	,
;	space			

You can use a hyphen and underscore in file names, and except in cases where DOS or a command requires an extension, having an extension is optional (for example, *my-junk.txt* or *my_junk*).

Reserved File Names and Extensions. Besides not using certain characters, you cannot use certain reserved file names and extensions.

The reserved file names set aside by DOS include:

AUX	*CLOCK$*	*COM1*	*COM2*	*COM3*	*COM4*
CON	*LPT1*	*LPT2*	*LPT3*	*NUL*	*PRN*

Reserved extensions include:

BAK	Appended by BACKUP commands to the backup files.
BAT	Given to DOS batch files (executable scripts).
CHK	Appended to files recovered by CHKDSK commands.
COM	Given to executable program files.
EXE	Given to executable program files.
MAP	A default extension for the list files created by MS-LINK.
MSG	Application developed by HP have this extension.
OVL	DOS uses the extension for overlay files.
REC	DOS uses the extension for files recovered by RECOVER commands.
SYS	Given to files that are device drivers (programs that enable a system to communicate with a device).
$$$	Used by DOS to designate temporary files.

Some commonly accepted extensions include:

BAS	Given to programs developed in the BASIC language.
BIN	Given to binary files.
DOC	Given to text files that are documents.
TXT	Given to files containing text.

Wildcard Characters. In a file name or extension, DOS interprets * and ? as "wildcard" characters (that is, DOS substitutes characters for them).

*	means substitute an entire file name or extension.
?	means substitute a single character in a file name or extension at the relative location of the question mark. You can have a question mark for each relative character to be substituted (for example, *??1.wk?* specifies files whose name has two characters and one (1) and whose extension has *wk* and any character.

The following examples illustrate the process:

.	Specifies all file names and all extensions.
**.DOC*	Specifies all file names having the *DOC* extension.
Chapter?	Specifies all file names having *Chapter* and one additional character such as 1, 2, ... , 9 or A, B, ... , Z.
RPT.??*	Specifies all files having *RPT* and any additional characters, and also having a two-character extension.
???file.TXT*	Specifies files having any 3 initial characters, *file,* any additional characters, and also having the *TXT* extension.

By using wildcards, you can specify files very efficiently, but using them with commands such as ERASE, RESTORE, and BACKUP can produce disastrous or surprising results. A good procedure is to test a comprehensive wildcard specification with DIR to see what gets listed. Then, use the same specification with a command such as ERASE.

Parameters and Options (Switches)

Many DOS commands use or accept parameters and options that alter or specify how the commands work.

Parameters. Parameters such as *device* or *codepage* in a command imply that you can provide a value for a device or code page that determines how a command works. For example, substituting *PRN* for *device* would tell a command to use the parallel printer while substituting *COM2* would tell a command to use the device hooked to the second serial port.

Parameters can be required or optional. The syntax encloses optional parameters in brackets. For example,

[*junk stuff pieces*]

means that *junk, stuff,* and *pieces* are all optional parameters. The manual avoids using:

[*junk*] [*stuff*] [*pieces*]

because the syntax for some commands can become too cluttered to read.

Options (Switches). Besides parameters, some commands have options (also called switches) that alter the way a command works. Some switches stand alone and others have parameters. Switches always begin with a slash, and you either use them or do not use them (that is, they toggle an action). For example:

/X The backslash with a letter names a stand-alone switch that toggles an action a command can take. For example, such a switch might cause a command that normally ignores read-only files to include read-only files.

/X:junk This switch has a parameter. Consequently, deciding to use the switch will toggle an action, and substituting a value for the parameter will determine how the action works.

The syntax encloses options in brackets to imply that you can decide whether to use them, but the syntax does not enclose each switch separately.

Besides reading the syntax, the "Tips and Hints" section always discusses parameters and options, and explains how they work.

Using the Parts in a Command

Earlier sections discusses the parts of a command: the command name, drives and disks, devices, paths, directories, files, reserved names, extensions, wildcards, and parameters/options.

In particular, the parts of a command can include a drive, path, filename, and extension. Together, a drive, path, filename, and extension can be viewed as a file specification, or *filespec*. To help you read the syntax, this manual uses *filespec* to imply that you may need to specify a:

1. drive such as *C:;*
2. path such as *\proj\new\;*
3. filename such as *report_1;* or
4. extension such as *.doc.*

When at least one of these items is required, you see *filespec*. When any or all of the items are optional, you see [*filespec*].

In most cases, DOS commands that use file specifications require specifying the filename, and when you do not specify a drive or path, the command uses the current drive and path. Extensions are usually optional, but they can be helpful for specifying the files a command acts upon. The "Tips and Hints" section always provides additional information about specifying files, and in particular, the section points out exceptions in how a command works.

To get a first look at the parts, here is the syntax for a variation of the MODE command that shows the idea while having some distinctions:

MODE *device* CODEPAGE [/*STATUS*]

Study the syntax and note the following items:

MODE	The command name. Type this literally.
device	A required parameter (no brackets). Substitute a value such as *CON*.
CODEPAGE	A required parameter, but you do not substitute a parameter. CODEPAGE alters the function of MODE (the MODE command has 8 derivatives, and you determine the derivative with parameters such as CODEPAGE or CODEPAGE REFRESH). Another distinction is that you can just type CP for CODEPAGE. This illustrates a principle: For some DOS commands, you should study the syntax, examples, and tips/hints to get a complete picture of how the command works.
[/*STATUS*]	A switch that causes the command to display its status. The exception here is that the name of the switch is STATUS. In most cases, a single letter names a switch.
No filespec	This command does not have a file specification. Many other commands do have one or more file specifications.

As you can see, the syntax shows the structure of a command and the parts in it; but it does not show you what to type. You can deduce what to type by also studying the examples and tips/hints.

Using Symbols in Commands

The syntax of DOS commands can use the following symbols:

CAPS — Items in capital letters must be typed exactly as shown. The command is always in capital letters. Other entities appear in capital letters when necessary.

Italics — Items such as *filespec, codepage, /V* appear in italics to imply that you specify file(s), substitute values for parameters, and make decisions about using switches. For example, to enter a *filespec,* you might actually type `c:\reports\old\may.txt`

[items] — Brackets enclose similar items where you can optionally use the enclosed items according to the requirements for running the command. The following example is for a command having three switches:

[/V /C /P]

/ and \ — A slash (/) precedes options called switches (for example, */N:xx*). A backslash (\) has two meanings. It separates directories in a path; and designates the root directory when it is the first entry in a path (for example, *\docs\memos\cleanup.txt* has a file named `cleanup.txt` in a directory named `memos` in a directory named `docs` in the root directory).

| — A vertical bar separates two items when you can use one of the items, but not both.

... — An ellipsis implies that you can have more than one entry of the preceding type (for example, *filespec1 filespec2* ... means you might actually enter `rpt1.txt rpt2.txt rpt3.txt`).

Use punctuation such as colons, commas, and equal signs exactly as they appear in the syntax.

Using Internal and External Commands

DOS has internal and external commands.

Internal commands are loaded into memory when you start up DOS. Therefore, you never need to precede an internal command with a drive and path that tells DOS where to find the command.

External commands are not loaded into memory until you execute them. With few exceptions, the memory used by an external command is released when the command terminates. Because you can execute an external command from any location in the directory structure while the external DOS commands are typically stored in a certain directory, you may need to precede the command with a drive and/or path that shows DOS where to find the command.

Most reference manuals precede external commands with [*d:path*] to imply that you can optionally provide a drive and/or path to the command. This manual does not do this because we wanted to keep the syntax clean and easy to read.

You will, however, need to account for the location of external commands in your directory structure, and you have two ways to do this:

1. Most people put external DOS commands in the root directory on drive C. (Some applications expect DOS external commands to be in the root directory on drive C when you have a hard disk.) If you do this, and you work in subdirectories or on a disk in drive A, you need to specify *c:/* ahead on an external command.

2. An alternative is to have a PATH command in the AUTOEXEC.BAT file that runs when you start up DOS. (An AUTOEXEC.BAT file contains commands and programs that run when you start DOS or an application for the purpose of setting up DOS or the application.) By using PATH, you can tell DOS where to look for commands and programs. If you account for how you use your system, you never need to precede an external command with a drive or path.

Running External Commands With A Flexible Disk Drive and A Hard Disk Drive

People typically copy the files on the DOS system disk from drive A to the root directory on drive C during installation and then start up DOS from drive C. Then, depending on their system, people use drive A or drive B for data disks, applications, library disks, and so on (some system do not have drive B). In most cases, the root directory on drive C contains the DOS commands.

During a working session, if you move to a subdirectory on drive C or move to drive A, and execute an external command, you need to specify *c:* ahead of the command to tell DOS the location of the command. For example,

1. If you are in *c:\utils\tools;* and
2. You want to format a flexible disk in drive A; and
3. The root directory on drive C has the DOS commands; then
4. You would need to type:

```
c:\format a:
```

   to help DOS find the FORMAT command.

You can avoid needing to do this by including an appropriate PATH command in the AUTOEXEC.BAT file for DOS. But whatever you do, the point is this: when DOS cannot find an external command, nothing serious happens. You merely get the following error message:

```
Bad command or filename

Press any key to continue
```

On getting this message, you need to retype the command, preceding the command with a drive and path to the command's location in the file system.

Running a DOS Command

This section shows how to use some entities and symbols to execute a DOS command. For convenience, the illustration uses a hypothetical command named DIG which has equally hypothetical entities.

```
C>A:\DIG DEEP C:\KIDS\NEED\MONEY.NOW /A=250 [Return]
```

The following items discuss the entities:

C> — The DOS prompt. You do not type this. It appears on the display to indicate that DOS is waiting, ready for you to type a command.

A: — Assuming DIG is an external command, you often need to type a drive and path that shows DOS where to look for a command. When and how you do this depends on how you set up your system and your location in the system when you execute an external command. In this case, you tell DOS to look for DIG in the root directory on drive A.

DIG — The name of the command. You never need a preceding drive or path for internal commands. You may need a preceding drive and/or path for external commands.

DEEP — A parameter that shapes how DIG works. The type, location, and nature of parameters to a command vary among commands. Some commands have no parameters. Others may have several.

C:\KIDS\NEED The drive and path names that show DOS where to find a file. Some commands read, write, or load a file; and when they do, you often need to specify a drive and path.

MONEY.NOW The name of a file, with its extension, that will be used in some way by the command.

/A=250 A special option called a switch that turns the feature determined by */A* on or off. A slash always precedes a switch, which usually appears at the end of a command line.

Return You always terminate a DOS command by typing the Return key, which is the same as the Enter key. Later, in the reference material, we do not show Return, but you always type it after you type a command.

Except for typing the command and Return, and perhaps typing a drive and path for an external command, all other entities in a command line can be required or optional, depending on the command and how you want the command to work. When using a command, think about what you want to do, study what the command does, and use the entities that make the command work according to your needs.

2

MS-DOS Commands

This chapter contains reference material for DOS 3.3 commands. Chapter 3 describes commands for processing batch files. The alphabetized commands in this chapter provide the information shown in the table. (To recover from errors, note the displayed error message, move to *Appendix A*, and apply the information provided in a box under the message.)

Information Provided for DOS Commands

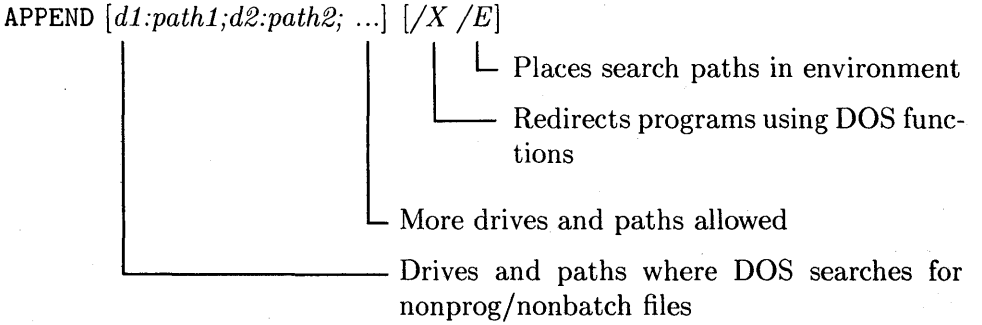
Type of Information	Description
Heading	The full name of the command and any short versions.
Version and Source	Lists versions of DOS having a command (for example, *Version 2 to present* means the command began with version 2 of DOS and appears in subsequent versions. The *Source* indicates whether a command is *Internal* or *External.*
Purpose	States what the command does (implies how to use it).
Syntax	Shows the command name and any options, specifications, switches, or parameters. The syntax does not indicate a drive and path for external commands, assuming instead that you provide a drive and path or use PATH according to your system. Brackets enclose optional items, and a "Tips and Hints" section explains them.
Examples	Provides examples with descriptions.
Tips and Hints	Provides information about limitations, requirements, relationships, dependencies, and so on. You also get information about how and when to use the command.

APPEND

External, Version 3.3

Sets a search path (or paths) to one or more directories for data files. It also displays and deletes existing search paths. The command has several purposes, depending on how you use the syntax. (See "Examples".)

Syntax



Examples

Command	Result/Action
`append c:\tools;c:\util`	Makes DOS look for nonprogram and nonbatch files in *\tools* and *\util* on drive C.
`append;`	Cancels existing APPEND search paths.
`append`	Shows the current search path set by APPEND.
`append /x /e`	The */X* makes DOS intercept SEARCH FIRST, FIND FIRST, and EXEC function calls from DOS. With */E*, DOS places the paths given to APPEND in the environment. Do not specify a path with these switches.

Tips and Hints

- APPEND loads from a disk the first time you use it. Thereafter, it is a memory-resident command until you restart DOS.
- You can use */X* and */E* only when you first execute APPEND. You must not specify any path names with these switches.
- Not entering drive names uses the current drive.
- When you do specify paths:
  - Semicolons must separate the paths.
  - A search for data files occurs in the order of paths specified by APPEND, until the list of directory paths is exhausted.
- The specified paths cannot exceed 127 characters. DOS skips invalid paths.
- The */X* switch enables executable files that use EXEC, SEARCH FIRST, and FIND FIRST functions to access search paths. The executables include DOS commands such as COMP, DIR, and TREE. If you don't specify the switch, only the executable files that use OPEN FILE, HANDLE OPEN, and GET FILE SIZE can access the APPEND search paths.
- *APPEND;* cancels existing search paths.
- When using */E,* DOS establishes an environment variable named *APPEND.* The variable holds the paths set by an APPEND command.
- A file found by APPEND can be read safely by the program, but changes in the file are saved in a copy placed in the current directory, leaving the original file intact.
- To use APPEND and ASSIGN, you must execute APPEND before using ASSIGN.
- Do not use *APPEND /X* with RESTORE or BACKUP.

ASSIGN

External, Version 2 to present

Reassigns all disk read and write requests from one drive to another drive.

Syntax

`ASSIGN [`*d1=d2 ...*`]`

- *d1=d2 ...* — Means you can have more than one assignment
- *d1* — Drive you want DOS to use instead of the usual drive
- *d2* — Drive that programs and DOS use when not assigned

Examples

Command	Result/Action
`assign b=c a=c`	Assigns drives B and A to drive C.
`assign`	Cancels assignments.

Tips and Hints

- You seldom need to use ASSIGN on a system having recent software and a hard disk drive. You typically use the command with software that recognizes only drives A and B.
- Do not assign a drive to a drive that is not on your system.
- You can assign a drive to another drive, but you cannot assign a drive to itself.
- Do not assign a real drive to a nonexistent drive.
- One command can have multiple assignments, separated by a space. For example:

```
c:\util\assign a=c b=c d=c
```

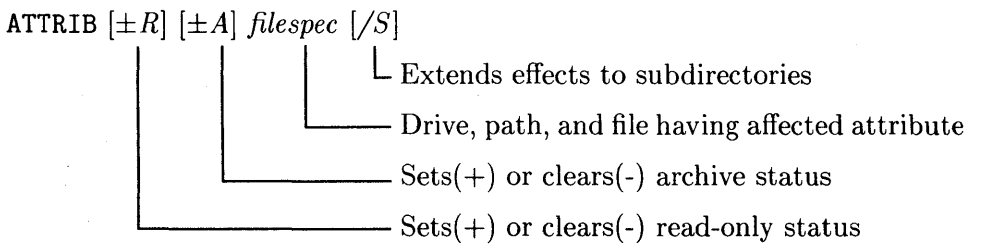
  where, in this case, DOS looks for ASSIGN in *c:\util* and assigns drives A, B, and D to drive C.
- Copy protection schemes used by some programs ignore ASSIGN.
- For a command such as *ASSIGN A=F,* which uses F or a "greater" letter, you must have previously set up an appropriate LASTDRIVE command in CONFIG.SYS.
- You can use a space between the drive letters, the = is optional.
- DISKCOPY and DISKCOMP ignore drive reassignments.
- FORMAT fails when you attempt to format an assigned drive.
- Cancel assignments before you use BACKUP, JOIN, LABEL, PRINT, RESTORE, or SUBST.

ATTRIB

External, Version 3 to present

Displays, sets, or clears the read-only and archive status for a file (or files).

Syntax



Examples

Command	Result/Action
`attrib +r a:\data.* /s`	Sets read-only status for files in the root directory on drive A named *data* with any extension. In addition, the command applies the status to all subsequent subdirectories (version 3.3 only).
`attrib -r -a a:\*.*`	Clears the read-only and archive status of all files in the root directory on drive A.
`attrib +r -a c:\rpt?.txt`	Sets read-only status and clears archive status of files in *c:* whose names have *rpt* and one additional character plus the *txt* extension (for example, *rpt1.txt*, *rpta.txt*, and so on).
`attrib report`	Displays the attributes of the file named *report*.

Tips and Hints

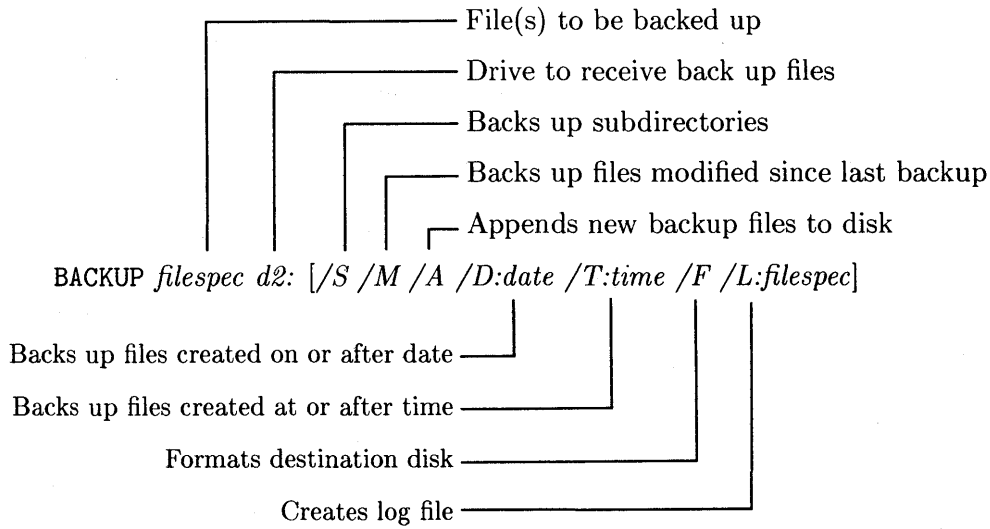
- ATTRIB works best when you use supplemental files such as library files. Just make the files read-only, and you will not need to worry about writing over them.
- You cannot modify or delete a read-only file, but you can rename such a file. The latter practice is not recommended because you might confuse programs that use the file.
- Not specifying a drive or path uses files in the current drive and path.
- Not specifying *filespec* causes an error.
- You can use both options in a command, mixing the signs as needed (for example, *+R -A*). But do not set an option to on and off (for example *+R -R*).
- Copying read-only files does not pass on the read-only attribute to the new file.
- The archive attribute affects BACKUP and XCOPY when you use then with */M*. A file is copied when the archive attribute is set and is not copied when the attribute is not set.

BACKUP

External, Version 2 to present

Backs up at least one file from a disk onto another disk. (Note: Using this command appropriately is important. Read all the information before doing a backup.)

Syntax



Examples

Command	Result/Action
`backup c:\*.* a:` `backup c:\ a:`	Both commands back up all files in the root directory on drive C onto the disk(s) in drive A. If necessary, BACKUP prompts you to insert the next disk into drive A. You must keep track of the order of the disks.
`backup c:\rpts\*.doc a: /s`	Backs up all files and directories in *c:\rpts* having any file name and the *doc* extension. The command places the files and subdirectories on the disk(s) in drive A.

Switches in the Syntax

/S Backs up all subdirectories beginning at the current or specified source directory.

/M Backs up all files modified since the last backup. The switch also backs up files that have been created since the last backup.

/A Adds the specified source files to previously backed-up files. Without the switch, DOS deletes the files already on the backup disk (that is, it deletes all files on a flexible disk or in */BACKUP* on a hard disk). To use this switch with flexible disks, the initial disk must contain the special files created by BACKUP during the previous backup. The files are called *CONTROL.xxx* and *BACKUP.xxx* where *xxx* is the number of the disk. Saying this another way, the first disk for the current backup should be the last disk from previous backup.

The switches continue on the next page.

/D:date Backs up files created on or after *date*. The format of *date* varies, depending on the continent. Also, the delimiter can vary (for example a slash).

mm-dd-yy	North America
dd-mm-yy	Europe
yy-mm-dd	East Asia
mm	The month (1 through 12)
dd	The day (1 through 31)
yy	The year (00 through 99)

/T:time Backs up files changed or created at or after *time* (for example *2:30:45*) on the date given by */D:date*. Some countries use a period as a separator instead of a colon. Do not use this switch without using the switch for date.

hh	The hour (0 through 23)
mm	The minutes (0 through 59)
ss	The seconds (0 through 59)

/F Formats the destination disk before backing up files on it when the disk is unformatted (version 3.3 only). Including this switch lets you use unformatted disks for the backup. (The formatting slows down the back-up process.)

The switches continue on the next page

/L:filespec Creates a log file according to these rules:

- Not giving a drive uses the source drive. You cannot use the destination drive.
- Not giving a path uses the current directory.
- When giving a logfile name, use a colon between *L* and any additional specification. Do not use spaces.
- When not giving a logfile name, do not include a colon. BACKUP will create *BACKUP.LOG*.
- If a logfile exists, BACKUP adds to it. If one does not exist, BACKUP creates a new file.
- If you use the switch and do not specify anything, BACKUP aborts.

Tips and Hints

- You must specify a source and destination drive. Neither drive can be used in ASSIGN, JOIN, or SUBST commands. Not specifying a drive causes an error. For the destination, you specify only the disk drive (for example, *A:*).
- Besides the drive:
  - You can specify a path and/or file(s) for the source.
  - Not specifying a path backs up files in the current directory.
  - Not specifying a file or files backs up all files in a directory.
  - You can use wildcard characters (* and ?).
- You can use the */F* switch to format the disks during the backup.
- You can get several messages concerning a full disk. A hard disk must have enough room to hold all of the backed up files. If you don't have enough room, you will need to delete files or otherwise provide for more storage space, and try again.
- When you back up onto flexible disks, DOS asks you to insert a new disk after the current disk fills up. Insert an appropriate disk (not one having

valuable data) and be sure to record the order of the disks. To restore a backup later, you must insert the disks in order.

- You can retrieve a backup only with RESTORE. Otherwise, you cannot use the backup files.
- BACKUP follows the convention provided by VERIFY. If VERIFY is ON, DOS ensures correct storage of the files. IF VERIFY is OFF, DOS does not ensure correct storage.

This page has no reference material.

BREAK

Internal, Version 2 to present

Determines when DOS looks for a CTRL-Break or CTRL-C to stop a program.

Syntax

```
BREAK [ON | OFF]
      └── Turns BREAK on or off
```

Examples

Command	Result/Action
`break on`	Turns BREAK on so DOS will look for CTRL-Break.
`break`	Displays the current status of BREAK.

Tips and Hints

- BREAK is either on or off.
- When BREAK is on, DOS looks for CTRL-Break when performing any DOS operation. When BREAK is off, DOS looks for CTRL-Break only when performing an operation with the:
  - Terminal or keyboard.
  - Printer.
  - RS-232 adapters (asynchronous adapters).
- DOS displays no messages when BREAK is on.
- DOS works about 2% slower when BREAK is on. The command has little effect on disk performance.
- A BREAK command has the same effect as having an equivalent BREAK directive in CONFIG.SYS.
- You seldom need to have BREAK on unless you run long programs that have little interaction with the monitor, keyboard, and printer. Some programs bypass DOS for character and disk operations. When these programs are running, CTRL-Break or CTRL-C may not interrupt them even though a BREAK ON command is in effect.

CHCP

Internal, Version 3.3

Changes the code page (font) used by DOS for as many devices as possible.

Syntax

CHCP [*codepage*]

└── A 3-digit code page number

Examples

Command	Result/Action
chcp 437	Changes the code page to the font for the United States.
chcp	Displays the current code page. For example: `Active code page: 850`

Tips and Hints

- Valid values for *codepage* include:

United States	*437*
Multilingual	*850*
Canadian	*863*
Denmark/Norway	*865*

- After successfully selecting a code page, the new code page remains active until you change it or restart DOS. If you have directives in CONFIG.SYS for devices that use code pages (for example, *DEVICE=PRINTER.SYS*), CHCP loads the code pages for the devices.
- CHCP sets all possible devices to use a font. The MODE CODEPAGE SELECT command is similar, but changes only one device per command.
- If you don't specify the location of COUNTRY.SYS when you execute NLSFUNC, COUNTRY.SYS must be in the root directory of the current disk.

CHDIR or CD

Internal, Version 2 to present

Changes the current directory to a specified directory, or shows the path of the current directory.

Syntax

Changing the Directory

CHDIR [*d:*][*path*]

└── The drive and path specifying the new directory

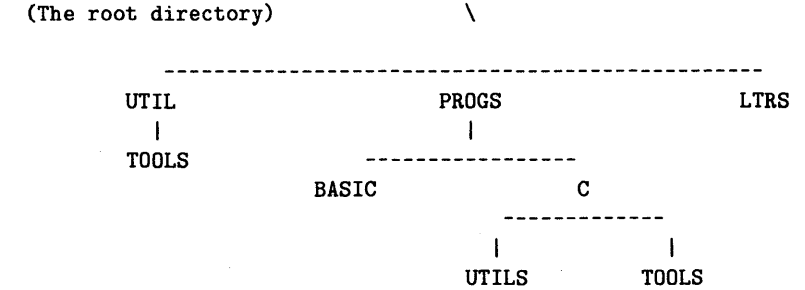
Showing the Current Directory

CHDIR [*d:*]

└── The drive having the directory you want to show

Examples

For the examples, assume the following directory structure on drive C:



Command	Result/Action
`chdir c:\util\tools`	Changes the directory to *TOOLS* under *UTIL* under the root directory on drive C, using an absolute path.
`cd c:`	Displays the current directory on drive C (for example, `C:\PROGS` where *PROGS* is the current directory).
`cd basic`	Changes the directory to *BASIC*, which must be under the current directory (*PROGS*) on the current drive (*C:*), using a relative path.
`chdir ..\ltrs`	Moves to the next higher directory on the current drive, and changes the directory to *LTRS*.

Tips and Hints

- Absolute paths start with a backslash. You can then specify subdirectories in the path as desired (for example, *\RPTS\OLD\JUNE* proceeds absolutely to *JUNE*).
- Relative paths do not start with a backslash. They assume the current directory and move down to a subdirectory. For example, if you are in *PROGS*, then *C\UTILS* is a relative path to *UTILS*.
- By using ..\ or ..\..\ (and so on), you can move up one, two, or more levels in a relative sense and then specify a directory (for example, `..\..\docs`).
- A specified path can have up to 63 characters.
- Using an invalid path displays an error message.

CHKDSK

External, Version 1 to present

Checks the directories and file allocation table (FAT) of a disk. Reports disk and memory status. Can repair errors in the directories and FAT.

Syntax

```
CHKDSK [filespec] [/F /V]
                  |   |
                  |   └── /F makes repairs; /V displays messages
                  |
                  └────── Drive, path, and file name for files being checked
```

Examples

Command	Result/Action
`chkdsk c:\*.* /F /V`	Checks all directories and files in the root directory on drive C. The */F* (for fix) asks CHKDSK to make possible repairs. The */V* (for verbose) asks CHKDSK to display messages and show progress during the process.
`chkdsk a:`	Checks all directories and files on the disk in drive A. The command does not fix anything and does not show messages.
`c:\chkdsk c: /V >prn`	Checks directories and files on drive C with the verbose mode and redirects messages to the printer.

Tips and Hints

- Not specifying a drive or path uses the current drive and directory.
- If you have a one-drive system and want to check a disk, insert a system disk in drive A and execute:

```
chkdsk b:
```

By specifying drive B, you are asked to insert the desired disk.

- If you include a file specification, CHKDSK checks the file for continuity.
- You must use compatible versions of DOS and CHKDSK because the command has been changed across versions.
- CHKDSK will not work on disks affected by an ASSIGN, JOIN, or SUBST.
- When you run CHKDSK with */F* and the command finds a problem, the command displays a message and asks you about repairing it. The messages can be unclear, and in some cases, you need expert knowledge of DOS to decide what to do.

CLS

Internal, Version 2 to present

Clears the current display, leaving the cursor at home (the upper-left corner).

Syntax

```
CLS
```

Examples

You do not need any. Just execute *CLS*.

Tips and Hints

- If you set the foreground and background colors with ANSI control codes, the color settings remain in effect.
- If you did not set the foreground and background colors, the display shows light characters on a dark background.
- CLS has no effect on DOS; it affects only the display.

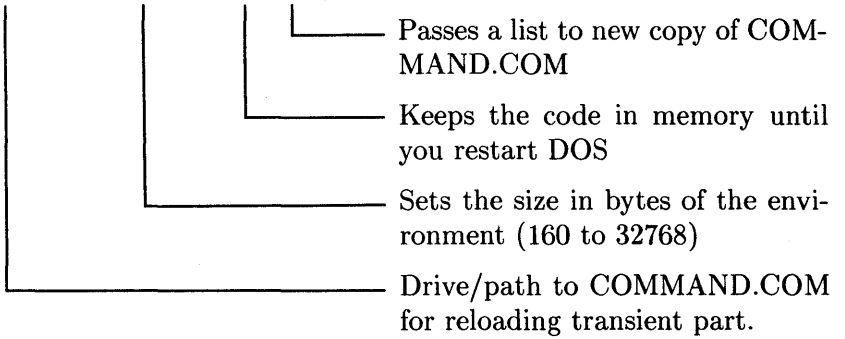
COMMAND

External, Version 2 to present

Runs a second copy of COMMAND.COM, which is the command processor.

Syntax

COMMAND [*d:*][*path*] [*/E:size /P /C string*]



Examples

Command	Result/Action
`command /C batch-x`	Establishes a new command processor (COMMAND.COM) that contains the commands contained in *batch-x*.

Tips and Hints

- Provide the drive and path when COMMAND.COM is not in the root directory of the startup disk.
- The *string* used with the */C* option is interpreted by the copy of the new COMMAND.COM as though you typed *string* at the system level. If you use */C*, it must be the last option. Do not use the */C* and */P* options in the same command.
- COMMAND.COM is the only processor that can be loaded into memory. This has several implications:
    - Executing COMMAND with no parameters loads a new copy of the command processor, but the environment remains as it was.
    - You can use *COMMAND /C batch_name* to pass control between batch files that make DOS behave according to the contents of the files. When a batch file finishes, COMMAND exits and returns control to the first batch file.
    - You can use COMMAND with */C* and a DOS command such as CHKDSK.
- If you execute COMMAND with */P*, you can use EXIT to get back to the previous processor level.
- Running programs from a secondary command processor is called running programs from a *shell.* Menus for applications are often run from DOS shells.
- In using */E:size*, the size can range from 160 to 32768 in multiples of 16 bytes. Changing the environment of a secondary processor has no effect on the environment of the primary processor.

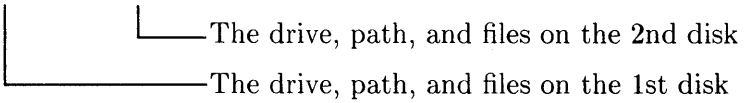
COMP

External, Version 1 to present

Compares the contents of two files and reports differences.

Syntax

COMP [*filespec1*] [*filespec2*]



Examples

Command	Result/Action
`comp c:\*.* a:\*.*`	Compares all files in the root directory on drive C with all files in the root directory on drive A. (For context, suppose you copied all files in the root directory on A to the root directory on C and you want to compare them.)
`comp *.txt *.doc`	Using files from the current directory and drive, the command compares files having the *txt* extension with files having the *doc* extension.
`comp`	Without parameters, the command prompts you to specify the parameters. Trying the command this way provides a good training session.

Tips and Hints

- Not specifying anything makes COMP prompt you for information about the primary and secondary files.
- Not specifying drives and paths uses the current drives and paths.
- Not specifying a file name for the primary file uses all the files in the specified or current primary directory. In the secondary directory, the

command compares only the files matching the files from the primary directory.

- COMP compares normal disk files, it does not examine hidden or system files.
- COMP does not compare files having matching names but different lengths. You get a message saying this.
- COMP ends comparing and aborts on finding 10 pairs of files that do not match.
- The drive, path, or file specifications can be the same for the primary and secondary file sets. You can use wildcard characters in the file names.

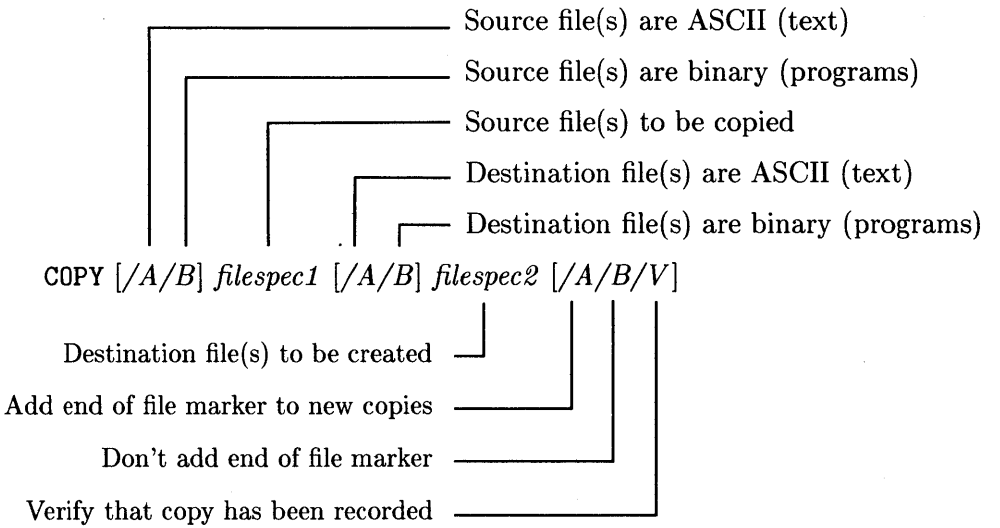
COPY

Internal, Version 1 to present

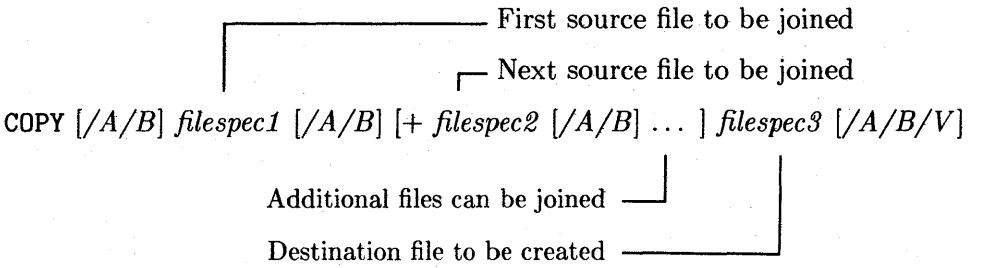
Copies one or more files from one disk or directory to another. COPY can also copy a file in a directory to the same directory provided the copy has a different name. Finally, COPY can combine two or more files into one file.

Syntax

Making Copies of Files



Joining Several Files to Make One Copy



See the "Tips and Hints" section to see how the drives, paths, filenames, and switches work. Except for specifying files, you have many options for executing a COPY command.

Examples

Command	Result/Action
`copy a:\*.* c:\app\*.*`	Copies all files from the root directory on drive A to the *app* directory in the root directory on drive C. (In general, you use this example to copy files from an application on a flexible disk in drive A into a directory of your choice under the root directory on your hard disk.)
`copy junk1 + junk2 my-junk`	In the current directory and drive, the command joins *junk1* and *junk2* into a file named *my-junk*.
`copy /a c:\*.doc b: /a`	Treats all files having the *doc* extension in the root directory on drive C as ASCII files (the first */A*). Copies the files to the root directory on drive B, giving the files an end-of-file marker (the second */A*). The typical EOF is Control-Z.

Tips and Hints

- Specify the options (switches) affecting the source ahead of the source specification. For example, */B c:\bin*.** makes the source consist of all files in the *bin* directory in the root directory on drive C.

  /A Treat the source files as ASCII (text) files. A command copies information up to but not including the EOF. (Without a specification, COPY assumes the source files are ASCII files.)

  /B Treat the source files as program (binary) files; which will copy the EOFs and treat them as normal characters.

- Specify the options (switches) for the destination files after the source specification. For example, *A:\RPT?.TXT /A* makes a copy in the root directory on drive A of all files having *RPT* and one character plus the *TXT* extension. You need the */A* after the destination files to add an end-of-file marker to the copies when you use */A* with the source files.
  - */A* means add an end-of-file marker to the copied files.
  - */B* means do not add an end-of-file marker to copied binary files.
  - */V* means verify that a copy has been recorded.

- The position of the */A* and */B* options determines their meaning. An option affects the file immediately preceding the option and all subsequent files until another, similar option is encountered. When using an option before a file specification, the option stays in effect until contradicted by another */A* or */B* option.

- When copying files having a source and destination specification:
  - Enter the source and then the destination specifications.
  - Unless specified, COPY uses the current drive and path.
  - Not specifying the destination name uses the source name.
  - You can substitute a device name such as *PRN* for the source or destination name.
  - When copying between disk drives, COPY assumes binary files as if you used */B*.
  - When copying to or from a non-disk device, COPY assumes ASCII files as if you used */A*.
  - Using */A* or */B* overrides the above two items.

- When copying files having only a source specification:
  - The source specification must have one or both of:
    1. a file name
    2. a drive or path name (or both). If you give both, at least one must differ from the current directory and drive.
  - The source cannot be a device name.
  - The destination is the current directory and drive.
  - COPY assumes binary files are being copied as if you used */B*.
- When joining (concatenating) files:
  - The destination file is the last file in the list unless a *+* precedes it.
  - Not giving a destination file uses the first file name in the source.
  - You must give a valid source file name; and all succeeding files must be preceded by a *+*.
  - You can have only one destination file specification. Not using wildcards in it creates one destination file. Using wildcards in it creates more than one destination file.
  - Not specifying a destination file uses the name of the first source file as follows:
    - The files being joined are appended to the first source file.
    - If you use wildcards, the destination file name has the name of the first source file.

CTTY

Internal, Version 2 to present

Changes the standard input and output device to an auxiliary console, or changes the input and output device back from an auxiliary console to the keyboard and video display.

Syntax

CTTY *device*

└ Name of the new input/output device (e.g. COM1, CON:)

Examples

Command	Result/Action
`CTTY COM1`	Makes the device attached to COM1 the new console. The device must be a terminal or teleprinter. After executing the command, DOS gets normal input from COM1 and sends anything meant for the video display to COM1.
`ctty con:`	Makes the keyboard and video display the console. The colon after a device name is optional. (In effect, the command cancels the above example.)

Tips and Hints

- You must specify a character device that can receive input from and send output to your computer system. You cannot use CTTY with a printer, for example, because it only accepts output. If you accomplished this, the system would wait forever for you to type commands on the printer's nonexistent keyboard. You would need to restart DOS.
- Some programs do not use DOS for standard input and output routines. Such programs bypass DOS and communicate with the hardware directly, or they use ROM BIOS I/O routines. CTTY has no effect on these programs.
- The command was added to DOS so people could use a terminal or teleprinter for input and output instead of the normal keyboard and video display.

DATE

Internal, Version 1 to present

Displays the system date, or lets you change the date.

Syntax

DATE [*date_string*]

└── A string for the date (e.g. 3-25-88)

Examples

Command	Result/Action
`date 12-9-88`	Sets the date to December 9 1988 in North America.
`date 6-10-89`	Sets the date to 6 October 1989 in Europe.
`date 88-3-9`	Sets the date to 1988 March 9 in East Asia.

Tips and Hints

- The format of the date varies among continents:

mm-dd-yy	North America
dd-mm-yy	Europe
yy-mm-dd	East Asia

  The ranges for the month, day, and year are:

mm	1-12
dd	1-31
yy	80-99

  Instead of *yy*, you can use *yyyy* where the range is 1980 to 2099.

- After starting DOS, you may see DATE and TIME prompts. Responding to the prompts lets you set the system date and time, or you can type [Return] to not set them.
- If you don't set the date, DOS might display and use an incorrect date.
- If the boot disk has an AUTOEXEC.BAT file, DOS does not display the prompts. You can include DATE and TIME commands in the batch file to set the system clock when you start DOS.
- Some computers have battery operated clocks. After setting the date and time when you install the system, you should not need to set them again unless the battery fails.
- The system time built into many PCs is a software clock. If you leave the system on all the time, the date is usually correct, but the time might not be correct.
- DOS uses the date and time when manipulating files and when you perform backups.
- Besides using a hyphen as a delimiter, you can use a period or slash (for example, *10-20-88, 10.20.88,* or *10/20/88*). The displayed date varies according to the country code set in CONFIG.SYS.

DEL

Internal, Version 1 to present

Deletes files from a disk.

Syntax

DEL *filespec*

└── The file(s) to be deleted

Examples

Command	Result/Action
`del a:*.*`	Erases all files in the current directory on the disk in drive A.
`del c:\tools\?ag.*`	Erases all files having one initial character and *ag* with any extension from the directory named *tools* in the root directory on drive C (for example, *tag.txt, sag.doc, gag.exe*).
`del junk.txt`	Erases *junk.txt* from the current directory and drive.

Tips and Hints

- When you specify *.*, DOS provides the following prompt:

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

  Be sure you know what such a specification will delete. The * wildcard can produce devastating results. Use DIR to check the files affected by a specification before using it with DEL.

- Like most commands that alter files, not specifying the drive and path uses the current drive and path. Not specifying at least one of a drive, path, or file causes an error.

- You cannot delete a directory, including the current directory and the parent directory, from a subdirectory. (Use RMDIR to remove a directory.)
- Technically, DEL alters the entries in a directory for the removed files and frees the space they occupied on a disk. If you decide you must get them back, and before you use the disk for any other operation, you can sometimes recover the files with special utility programs designed for this purpose.
- Specifying a path without a file name is identical to specifying a path with *.* for the file name. In the following example, DOS asks if you are sure, and deletes all files in the root directory on drive A if you enter Y.

```
DEL a:\
```

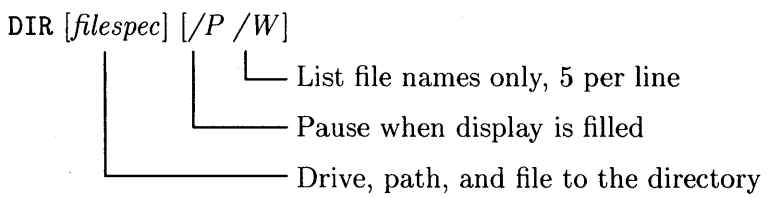
- The DEL command is identical in function to the ERASE command.

DIR

Internal, Version 1 to present

Lists files and subdirectories in a directory.

Syntax



Examples

Command	Result/Action
`dir c:\*.* /w /p`	Lists all files in the root directory on drive C. The */w* gives an 80 column listing that omits the size, date, and time related to a file. The */p* pauses the listing when the display fills up.
`dir /w >prn`	Lists files in current directory and drive in 80 column format and redirects the list to a printer.
`dir c:\projects\rpt?.* /p`	Lists files in the directory named *projects* in the root directory on drive C and lists all files having: *rpt;* one character after *rpt* (for example *rpt1*); and any extension. The */p* pauses the listing.
`dir c:\basic`	Lists files in *basic* in the root directory on drive C; showing sizes, dates, and times. Not using DATE and TIME to set the date and time makes the listing inaccurate.

Tips and Hints

- The options (switches) work as follows:

  /P Pauses a listing when the display fills up, waiting for you to press a key.

  /W Provides an 80 column list in which files appear in five columns per line; the dates and times do not appear; and subdirectories are not differentiated from files.

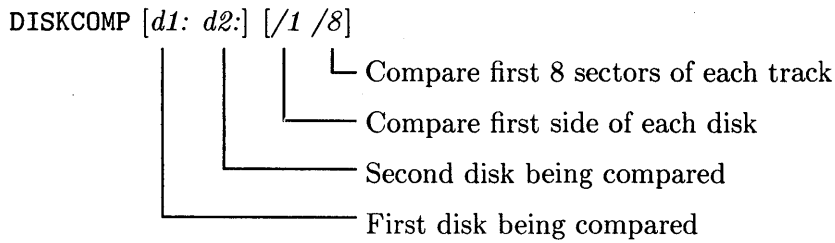
- DIR shows only the files in a specified or current directory. To see all files on a disk, use CHKDSK with /V or TREE with /F.
- You can see files in only one directory at a time.
- Not specifying a path or drive uses the current path and drive.
- Not specifying a file is the same as specifying *.*
- DIR is affected by an ASSIGN or JOIN.
- DIR can help you prevent accidentally erasing files. You can avoid losing them by first using DIR with the specification containing the wildcards. When the listing shows the files you want to erase, run DEL or ERASE with the tested specification.

DISKCOMP

External, Version 1 to present

Compares two flexible disks by tracks and sectors to determine if their contents are identical. (Be aware that two flexible disks can have the same files and still not be identical on a track/sector basis.)

Syntax



Examples

Command	Result/Action
`diskcomp a: b:`	Compares the disk in drive A with the disk in drive B.
`diskcomp a: a:`	Performs a single-drive comparison on drive A, prompting you to insert and change disks at appropriate times. (Use this command when you have only one drive.)
`diskcomp a: b: /8`	Compares eight sectors per track of the disk in drive A with eight sectors per track of the disk in drive B.

Tips and Hints

- The options (switches) work as follows:

  /1 Compare only the first sides of two disks.

  /8 Compare only 8 sectors per track, even if the first disk has 9, 15, or 18 sectors per track (DOS V2 and V3 formats).

- The main purpose for using DISKCOMP is to determine if a duplicated disk is compatible with an original disk. Because the command checks disks on a track/sector basis, and because DOS can copy files according to available space, you might not get what you expect.
  - Using DISKCOMP with disks on which you previously used COPY can suggest incompatibility when the two disks may contain the same files. (Do not use DISKCOMP for this. Instead, use COMP to compare disks duplicated with COPY.)
  - Using DISKCOMP to compare original and duplicate disks made with DISKCOPY should show whether a DISKCOPY command made an identical copy.
- Do not use DISKCOMP with disks involved in an ASSIGN, JOIN, or SUBST; or with a virtual (RAM) disk.
- When specifying drives:
  - Not specifying drives uses the current drives.
  - Specifying only one valid drive causes that drive to be compared to the current drive.
  - Specifying a valid hard disk drive or an invalid flexible disk drive causes an error.
  - When comparing two disks using only one drive, DOS prompts you to change disks.
- A duplicated disk is faulty when it was made with DISKCOPY and a DISKCOMP shows there is a problem. Just run DISKCOPY on the disks again. If there is still a problem, discard the duplicated disk, and try again with a new disk.

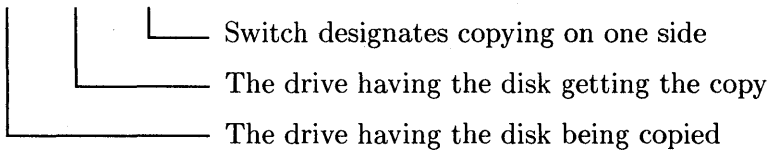
DISKCOPY

External, Version 1 to present

Copies the entire contents of a flexible disk to another flexible disk on a track/sector basis. (The command works only with flexible disks.)

Syntax

```
DISKCOPY [d1: [d2:] ] [/1]
```



Examples

Command	Result/Action
`diskcopy a: b:`	Copies everything on the disk in drive A to the disk in drive B.

Tips and Hints

- The */1* option makes the command copy on one side of a disk.
- Both disks must be flexible disks. You cannot use DISKCOPY with hard disks, virtual disks, or disks affected by a JOIN or SUBST command. Specifying an invalid disk causes an error.
- DISKCOPY ignores the effects of an ASSIGN command.
- Write protect the source disk to prevent losing valuable original disks. Using CTRL-Break or opening a drive door will not always prevent damage if you inadvertently do something wrong.

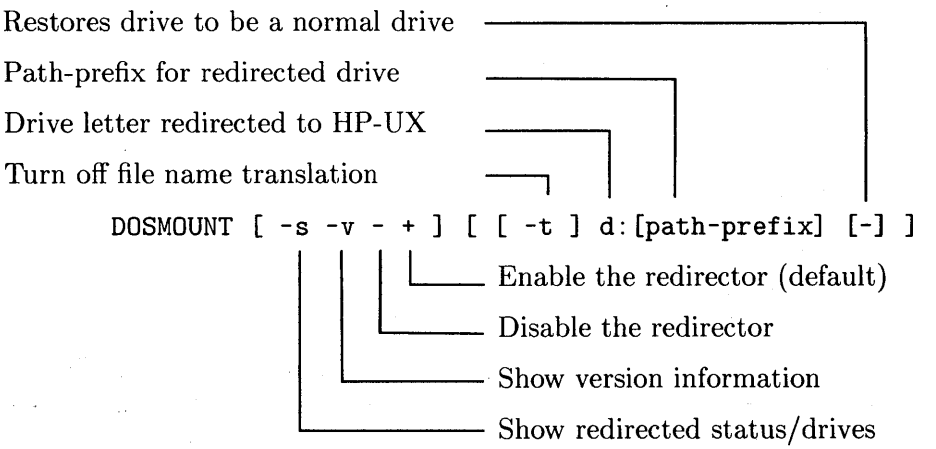
- When specifying drives:
  - Not specifying a source drive uses the current drive.
  - Specifying an invalid drive causes an error.
  - Specifying only one drive makes that drive the source and destination drives (for example, if you have only one disk drive).
  - Specifying only one drive for a system having two drives causes an invalid drive error.
  - Specifying a name twice is the same as specifying one drive.
  - A system having only one drive uses drive A regardless of your specification.
  - Prompts to change disks appear when you have only one drive.
- After finishing a copy, DISKCOPY prompts you about copying another disk. Answering *Y* lets you make another copy of a disk. Answering *N* stops the program and returns you to DOS.
- DISKCOPY automatically formats a destination disk that is not formatted or that has a non-DOS format, using the same format as the source disk. The formatting does not apply when copying a double-sided disk on a single-sided drive while using the */1* switch. It is better to format disks before using DISKCOPY. Do not use disks reported by FORMAT or CHKDSK as having bad sectors.
- DISKCOPY destroys information on the destination disk, the disk in the second drive.
- Running DISKCOMP on the disks after copying tells you if the copy is correct. (If you used COPY to make a copy, do not use DISKCOMP.)
- You must use compatible disks and disk drives. For example, using a 1.2 Mbyte disk with a 360 Kbyte drive causes an error.

DOSMOUNT

External HP extension, Version 3 to present, DOS Coprocessor Only

Redirects a drive letter for use in accessing the HP-UX file system.

Syntax



Examples

Command	Result/Action
`dosmount e:`	Redirect drive E to the HP-UX file system without resetting the path-prefix for E (if any).
`dosmount d: \usr\bin`	Redirect drive D to the HP-UX file system and set the path-prefix for D to *\usr\bin*.
`dosmount -t d:\usr\bin`	Redirect drive D to the HP-UX file system, set the path-prefix to *\usr\bin*, and turn off file name translation on drive D.
`dosmount f:-`	Restore drive F as a normal drive.

Tips and Hints

- You must run the REDIR command before your first use of the DOSMOUNT command.
- When setting up a redirected drive, you can use drive letters D through Z, subject to the value of the LASTDRIVE directive in CONFIG.SYS.
- You can specify only one redirected drive each time you run DOSMOUNT.
- When you do not specify the path-prefix, the redirected drive uses the HP-UX root.
- DOSMOUNT uses the HP-UX location set by the path-prefix as the root of the DOS redirected drive. If path-prefix does not exist in the HP-UX file system, a *File not found* error occurs when the redirected drive is accessed.
- Since path-prefix is an HP-UX path, it need not conform to conventions for DOS file names. You can use \ or / as directory delimiters. You must specify the *-t* option if a path-prefix contains upper-case letters.
- Use the *-t* option, which turns of file translation for a specified redirected drive, with HP-UX directories and files having DOS-compatible names. HP-UX file names that are not valid DOS names are ignored when using *-t*.

Tips and Hints continue on the next page.

- Concerning HP-UX file name conversion, if an HP-UX file name does not map directly into a valid DOS file name and you did not specify *-t* for the drive, the redirector generates a pseudo-name for the file as follows:
  - Extensions having 1 to 3 lower-case letters are preserved.
  - The first five valid DOS file-name characters are preserved. Non-valid characters are replaced by a tilde (~). The sixth character is a tilde. The seventh and eighth characters are coded indices to an internal cache. The following examples show the idea:

```
longfilename.exe ==> LONGF~xx.EXE
hello.exee ==> hello~xx
ACE ==> ACE~~~xx
ace ==> ACE
```

  - When a path begins with ~\ or \~\, the home directory is prepended to the path as determined by the HOME environment variable. Nothing is prepended when the HOME variable does not exist. The prepending action can create an invalid path, depending on the value of HOME.

This page has no reference material.

DOS2UX

External HP extension to DOS

Converts ASCII files from the DOS format to the HP-UX format. (Besides working in DOS, the command works in HP-UX.)

Syntax

`DOS2UX` *filespec*

└— The file being converted

Examples

Command	Result/Action
`dos2ux autoexec.bat >autoexec.ux`	Converts a DOS file named *autoexec.bat* to the HP-UX format and redirects the converted output to a file named *autoexec.ux*.
`type profile | dos2ux >c:.profile.ux`	The TYPE command in the pipeline sends the file named *profile* to DOS2UX for conversion, and the converted output from the DOS2UX command is redirected to a file named *profile.ux* in the current directory on drive C.

Tips and Hints

- HP-UX cannot read DOS ASCII files directly. To use a DOS file in HP-UX, you need to first use DOS2UX to convert the file. For example, to use *vi* to edit a DOS file, you should convert the file and then edit it.
- In most cases, DOS2UX works best with redirection and pipelines.
- Not specifying a file makes DOS2UX accept input from the standard input device, probably the keyboard.
- Unless redirected or piped, the converted output goes to the standard output device, probably the display.
- The original file that provides input is not changed.
- To get a more complete picture of converting DOS and HP-UX files, see the UX2DOS command.

ERASE

Internal, Version 1 to present

Removes a file or files from a directory. (Use the information for this command when using DEL.)

Syntax

ERASE *filespec*

└ The drive, path, file(s), and extension(s) to erase

Examples

Command	Result/Action
`erase a:*.*`	Erases all files in the current directory on the disk in drive A.
`erase c:\tools\?ag.*`	Erases all files having one initial character and *ag* with any extension from the directory named *tools* in the root directory on drive C (for example, *tag.txt, sag.doc, gag.exe*).
`erase junk.txt`	Erases *junk.txt* from the current directory and drive.

Tips and Hints

- When you specify *.*, DOS provides the following prompt:

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

  Be sure you know what such a specification will erase. The * wildcard can produce devastating results. Use DIR to check the files affected by a specification before using it with ERASE.

- Like most commands that alter files, not specifying the drive and path uses the current drive and path. Not specifying at least one of a drive, path, or file causes an error.

- You cannot erase a directory, including the current directory and the parent directory, from a subdirectory. (Use RMDIR to remove a directory.)
- Technically, ERASE alters the entries in a directory for the removed files and frees the space they occupied on a disk. If you decide you must get them back, and before you use the disk for any other operation, you can sometimes recover the files with special utility programs designed for this purpose.
- The ERASE command is identical in function to the DEL command.

EXE2BIN

External, Version 1.1 to present

Changes EXE files into COM or BIN files, provided the format is acceptable.

Syntax

`EXE2BIN` *filespec1* [*filespec2*]

- *filespec2*: Drive, path, and name of the COM file
- *filespec1*: Drive, path, and name of the EXE file

Examples

Command	Result/Action
`exe2bin junk.exe`	Creates a binary file named *junk.bin* from the executable file named *junk.exe*.
`exe2bin a:\junk.exe c:\utils\junk.com`	Changes *junk.exe* in the root directory on drive A into a COM file named *junk.com* in *utils* in the root directory on drive C.

Tips and Hints

- EXE2BIN converts executable programs (files having an EXE extension) to binary image programs (files having BIN or COM extensions). The binary programs take up less disk space and load faster. You can run into problems among versions of DOS by converting programs. Unless you use a compiler-based language, you probably do not need to use the command.
- When specifying drive, path, and file names, note the following items:
  - Not specifying a drive or path for the source file uses the current drive and path.
  - Not specifying a drive or path for the destination file uses the current drive and path.
  - You must specify a source file to be converted (that is a file having an EXE extension).
  - Not specifying a destination file name uses the name of the source file and adds the BIN extension. Specifying a destination file name and not specifying the extension uses BIN for the extension.
- The source file (the file having the EXE extension) must have a format that follows DOS conventions.

EXIT

Internal, Version 3.3

Exits the current command processor, and if a previous command processor exists, control returns DOS to that processor. (COMMAND.COM is the command processor for DOS).

Syntax

```
EXIT
```

Examples

Just run *EXIT* as appropriate.

Tips and Hints

- EXIT does not perform any operation when a previous command processor does not exist, it merely exits.

FASTOPEN

External, Version 3.3

Keeps information about hard disk directories and files in memory so DOS can quickly find and use files.

Syntax

```
FASTOPEN d:[=nnn ...]
```

- The dots mean you can specify several drives
- The number of directory entries held in memory (10 through 999)
- The name of the drive FASTOPEN affects

Examples

Command	Result/Action
`fastopen d:=40 c:=75`	Tracks 40 directories and files on drive D; 75 on drive C.

Tips and Hints

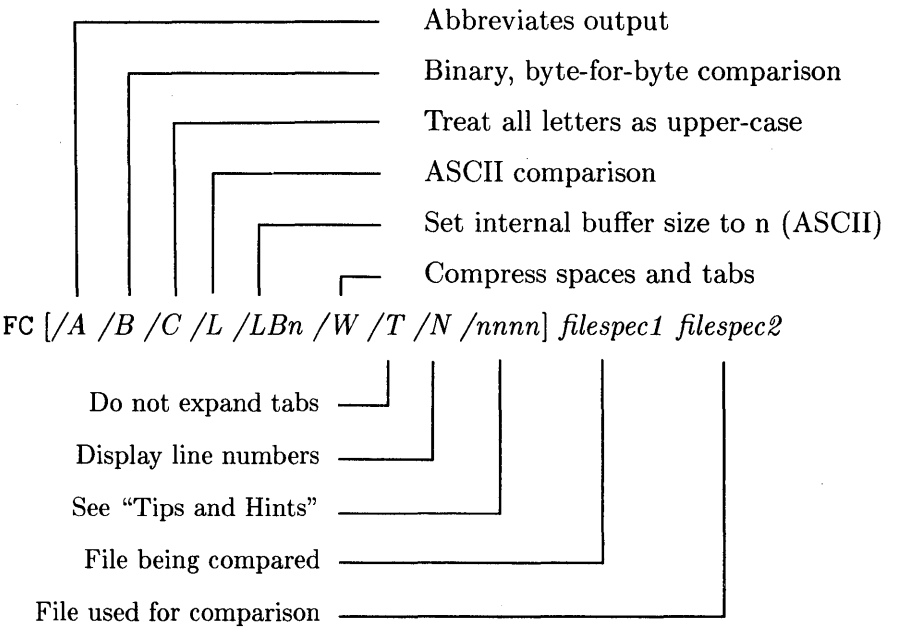
- When specifying drives, note the following items:
  - You cannot use FASTOPEN with flexible disk drives.
  - You can use FASTOPEN for several hard disk drives using the syntax for specifying drives and separating them with a space. The sum of the values for *nnn* over all specifications cannot exceed 999.
  - Not specifying *nnn* uses 10. If you specify a value, it should be greater than the deepest level of directories on the hard disk.
- When accessing a directory or file, FASTOPEN records its name and location. Then, on accessing a recorded directory or file, the access time is greatly reduced.
- FASTOPEN differs from BUFFERS. BUFFERS speeds up the system when DOS reads from or writes to a file, while FASTOPEN speeds up the system when you re-open a file.
- There is no best value for *nnn*. The default of 10 works well for most applications. Using a value greater than 200 can bog down the system. Try to use a value that slightly exceeds the number of subdirectories and files you use.
- After you install FASTOPEN, you might specify unacceptable drives or values for *nnn*. At this point, you must re-start DOS before trying FASTOPEN with different parameters.
- FASTOPEN uses about 40 bytes of memory for each directory and file it tracks.

FC

External, Version 3.3

Compares the contents of two files, which can be two text files or two binary files.

Syntax



Examples

Command	Result/Action
`fc /b c:\util\drill.bin a:\drill.bin`	Compares two binary files; *drill.bin*, which is in *util* in the root directory on drive C with *drill.bin*, which is in the root directory on drive A.

Tips and Hints

- The options (switches) work as follows:

  /A Abbreviates output from a comparison, showing only the first and last lines of each set of differences and placing an ellipses between the sets of lines.

  /B Does a binary comparison byte for byte, making no attempt to re-synchronize after a mismatch. Allowed extensions include: *EXE, COM, SYS, OBJ, LIB,* and *BIN.*

  /C Directs DOS to treat all letters as upper-case. Use only with ASCII files (files having only alphanumeric characters and allowable printing characters).

  /L Does an ASCII comparison on both files (this is the default option for files having extensions other than *EXE, COM, SYS, OBJ, LIB,* and *BIN*).

  /LHn Set the internal buffer to *n* during an ASCII comparison where the value of *n* must exceed the number of differing lines by 2. The value defaults to 100. Having too small a value causes FC to abort and you get an invalid comparison.

  /W Compresses white spaces causes by tabs and extraneous spaces. Use the option only with ASCII comparisons.

  /T Does not expand tabs (default treats tabs as 8 spaces).

The switches continue on the next page

- Continuation of switches:

  /N Displays line numbers. Use only with ASCII comparisons.

  /nnnn Specifies a number of lines that must match after a difference is found for the files to be considered as matching again.

- FC makes ASCII comparisons line by line. It makes binary comparisons byte by byte. A command compares the contents of the file specified first with the file specified second.
- Not specifying drives or paths uses the current drives and paths.
- For ASCII comparisons, output contains the following information:
  - File name
  - The last matching line preceding a difference.
  - The different lines.
  - The next matching lines.
- For binary comparisons, output contains the following information:
  - The first column shows the position in the files of each pair of mismatched bytes. The first byte in each file is byte number 0.
  - The second column shows the values of the mismatched bytes in *filespec1* in hex.
  - The third column shows the values of the mismatched bytes in *filespec2* in hex.
  - FC ignores any options entered after the file specifications.

This page has no reference information.

FDISK

External, Version 3.3

Configures a hard disk for use by DOS. During configuration, FDISK places partitions on the disk (a partition is an area for storing files).

Syntax

```
FDISK
```

Examples

Command	Result/Action
`fdisk`	Provides a series of menus that let you configure your hard disk.

Tips and Hints

- You must create a **primary partition** on every hard disk in which you store DOS files. The primary partition can have up to 32 Mbytes. Your system sees this partition as Drive C.
- You can create an **extended partition** on a hard disk. If the capacity of your disk exceeds 32 Mbytes, you must create an extended partition to use the additional capacity. The extended partition contains the rest of the drive. You can segment this partition into as many as 23 logical drives where each drive can contain up to 32 Mbytes. (You need not do this in the DOS Coprocessor environment, an HP-UX program does this for you.)
- The interactive menus provided by FDISK let you create primary and extended partitions. In addition, you can create logical drives, change the current partition, delete a partition, and display a partition.
- A logical drive acts like a separate hard drive. Logical drives are named E:, F:, G:, and so on up to 23 drives.

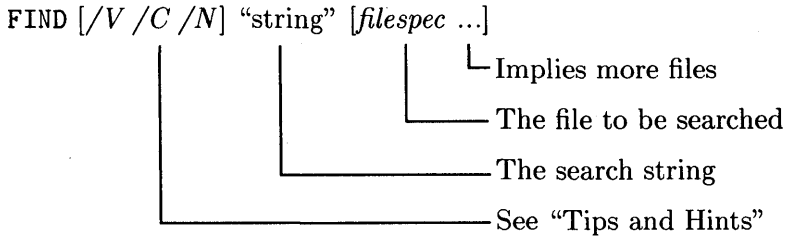
- The primary partition must be current to start DOS from a hard drive. But you may want to start your system with an operating system installed on the extended partition. To do this, use the "Change the Active Partition" menu.
- Deleting a partition destroys the information contained on it. Backup or move the files on the partition if you want to keep them.
- Displaying partition data provides information about its status (active-inactive), type (DOS-nonDOS), starting cylinder, ending cylinder, and size.
- Selecting "Next Fixed Disk Drive" moves you to the next physical hard disk on your system, if you have more than one.

FIND

External, Version 2 to present

Given a string and file, FIND displays the lines in the file that contain the string.

Syntax



Examples

Command	Result/Action
`find "Jones" ph-lst.txt`	Finds lines containing *Jones* in *ph-lst.txt*.
`find /n "Smith" a:\ph-lst.txt`	Searches *ph-lst.txt* in the root directory on drive A for lines containing *Smith* and displays the lines along with the line numbers.

Tips and Hints

- The options (switches) work as follows:

  /V Displays lines in a specified file **not** containing *string*.

  /C Counts the number of times *string* appears in a file and displays that value, but does not display the lines containing *string*.

  /N Displays the line number in front of each line in the specified file that contains *string*.

- When specifying a drive, path, and file, note the following items:
  - Not specifying a drive or path uses the current drive or path.
  - You can specify more than one file, separating each file with a space.
  - Not specifying a file causes FIND to expect input from the keyboard.
  - You cannot use wildcards in the file specification.
- Options must appear between FIND and *string*.
- You must enclose *string* in double-quotes (for example, *"John"*). To include double-quotes in the string, double quote them and the string (for example, *""John""*). Case is important. You can use numbers and special characters in *string*.
- You can redirect output to a suitable device. For example:

```
c:\find "503" b:\zip.txt >prn
```

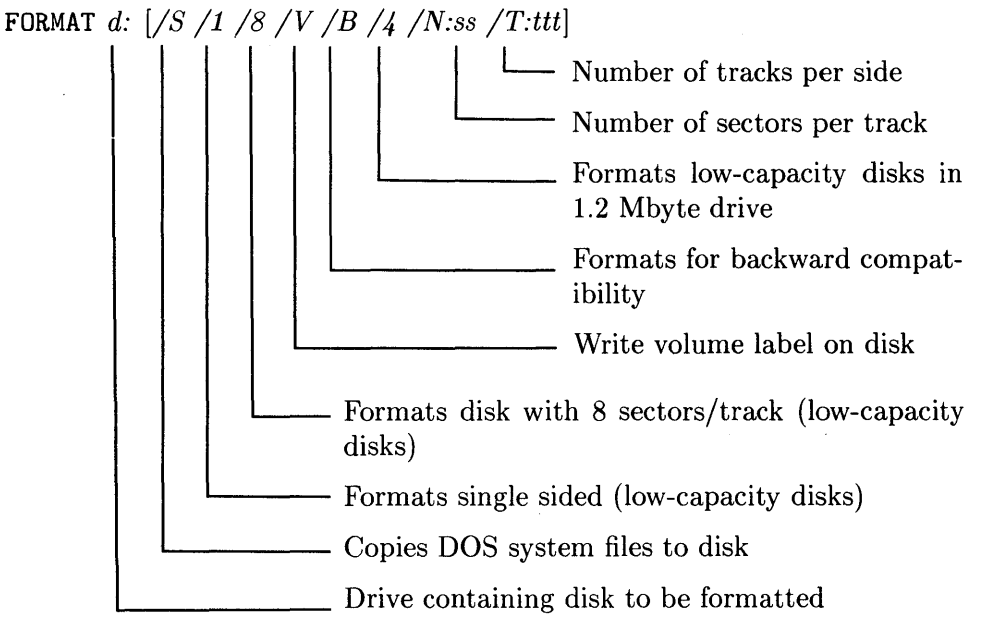
  finds all lines in a zip-code file having 503 and sends the lines to a printer.

FORMAT

External, Version 1 to present

Initializes a disk so it can contain files and be used by DOS. FORMAT checks a disk for defective tracks and can place DOS on a disk. (To format a 3.5 inch 720 Kbyte disk for use with an HP 150 computer, use FOR150 instead of FORMAT.)

Syntax



Examples

Command	Result/Action
`format a:`	Formats the disk in drive A. (This example shows how to format flexible disks used in drive A for storing data files.)

Tips and Hints

- The options (switches) work as follows:

/S	Copies the two hidden system files plus COMMAND.COM on a disk so you can start up DOS from the disk.
/1	Formats a single side of a 5.25 inch flexible disk (160/180 Kbyte). Do not use the switch with hard disks, high capacity flexible disks (1.2 Mbytes), or 3.5 inch flexible disks (720 Kbytes or 1.44 Mbytes).
/8	Formats a disk with 8 sectors per track (160/180 Kbyte single-sided disks or 320/360 double-sided disks). Do not use the switch with hard disks, high capacity flexible disks (1.2 Mbytes), or 3.5 inch flexible disks (720 Kbytes or 1.44 Mbytes).
/V	Writes a volume label to the disk. FORMAT prompts you to enter a volume label after formatting the disk.
/B	Formats an eight-sector flexible disk, leaving places in the directory for any version of DOS, but does not copy the operating system to the disk.
/4	Formats a single-sided or double-sided flexible disk in a 1.2 Mbyte drive.
/N:ss	The switch specifies the number of sectors per track. Use */N:9* to format a 720 Kbyte disk.
/T:ttt	The switch specifies the number of tracks per side. Use */T:80* to format a 720 Kbyte disk.

- The following options do not work together:
  - */V*, */N:ss*, */T;ttt*, or */S* with */B*
  - */V*, */N:ss*, or */T;ttt* with */8*
  - */1*, */4*, */8*, */B*, */N:ss*, or */t:ttt* with a hard disk drive.
- FORMAT destroys information left on previously formatted disks. (Do not format a disk containing useful files.)
- Not specifying a drive formats the disk in the current drive.
- Except when using a flexible disk as the destination disk in a DISKCOPY or when using a disk in a BACKUP with /F, you must format a new disk before you can use it.
- Unless directed otherwise by an option, DOS formats a disk to its maximum capacity.
- To make a flexible disk usable with all versions of DOS, use the */B* and */1* options. (This reduces the capacity of the disk to 140 Kbytes.)
- If you specify the */S* option when the current disk does not contain a bootable DOS, FORMAT prompts you to insert a DOS disk in the current drive before the formatting begins.
- When formatting a hard disk that has a volume label, DOS prompts you to enter the volume label. DOS aborts the formatting if you cannot enter the exact label.
- When formatting a hard disk not having a volume label, FORMAT prompts you about proceeding. Answering *Y* formats the disk, answering *N* aborts.
- The */4* option has no effect in the DOS Coprocessor or SoftPC environments. It is intended to format double-density flexible disks in a 1.2 Mbyte drive, which does not exist in these environments.
- Do not format any disk associated with a virtual disk drive.
- Do not format any disk affected by an ASSIGN, JOIN, or SUBST.

Exit Codes

FORMAT can display the following exit codes:

0	Last format successfully completed
1	Not defined
2	Not defined
3	Aborted by user via CTRL-Break
4	Aborted due to an error
5	Aborted due to an N response to a prompt

FOR150

External HP extension to DOS, Version 3.2 to present

Prepares a 720 Kbyte 3.5 inch flexible disk for use with the HP 150 computer.

Syntax

`FOR150` *d:* [*/V*]

- *d:* — The drive containing the disk to be formatted
- */V* — Prompts for writing a volume label on a disk

Examples

Command	Result/Action
`for150 a: /v`	Formats a 3.5 inch disk 720 Kbyte disk in drive A. Prompts for specification of a volume label.

Tips and Hints

- The command works only on 3.5 inch, double-sided disks intended for use in an HP 150 computer.
- The command cannot put the DOS system on a disk. A disk formatted with this command cannot be used to start DOS.
- A disk must have been previously formatted with a FORMAT command. You cannot format a new disk with a FOR150 command.

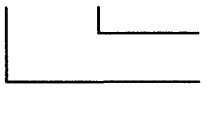
GRAFTABL

External, Version 3 to present

Loads tables into memory for additional character sets that can be displayed on the Color/Graphics Adapter (CGA) in graphics mode.

Syntax

GRAFTABL [*codepage /STATUS*]



Displays number of active codepage

3-digit number of the code page for the display

Examples

Command	Result/Action
`graftabl 863`	Loads the code page for Canada.
`graftabl`	With no specification, the command loads code page 437 (the United States).
`graftabl ?`	Shows the options for GRAFTABL.
`graftabl /status`	Displays the number of the current code page.

Tips and Hints

- Use only one of the two options per command (for example, do not use *graftabl 437 /status* in one command line).
- See MODE CODEPAGE for information about code pages.
- Load GRAFTABL to display characters in the ASCII range of 128 to 255 when you are in the all-points-addressable mode on the Color/Graphics Adapter.
- GRAFTABL has no effect when you are in text mode. Do not use the command unless you have a Color/Graphics Adapter being used in medium- or high-resolution mode.
- You must restart DOS to cancel GRAFTABL.
- The displayed exit codes include:

0	GRAFTABL installed successfully for the first time.
1	The code page was changed. When not specifying a new code page, there is an existing code page.
2	GRAFTABL is installed and no previous code page was installed.
3	Incorrect parameter, no change in GRAFTABL.
4	You have an incorrect version of DOS.

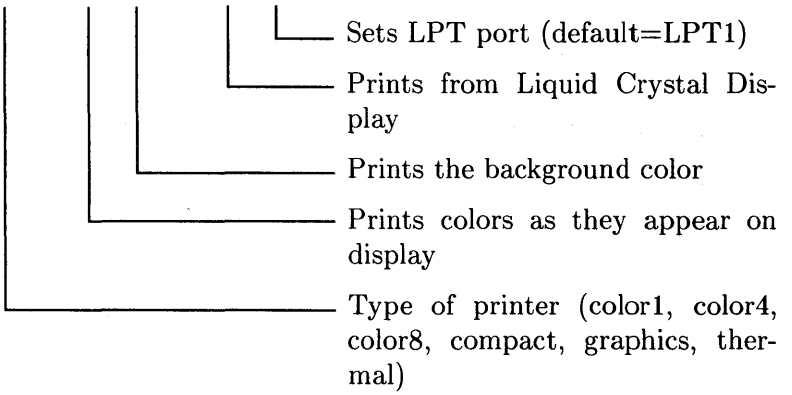
GRAPHICS

External, Version 2 to present

Prints the contents of a graphics screen on a printer when you use the Color/Graphics Adapter (CGA). (See note in "Tips and Hints")

Syntax

GRAPHICS [*printer*] [*/R /B /LCD /P=port*]



Examples

Command	Result/Action
`graphics /r`	The */R* makes the command print colors exactly as they appear on the display. After executing the command, type [Print Screen] or [Shift]-[Print Screen] to print the contents of the screen.

Tips and Hints

- NOTE: Do not use GRAPHICS with HP printers in HP mode. Instead, install and use the Print Screen Utility described in the *Using the Multiple Character Set Utilities* manual.
- After you run a GRAPHICS command, use [Print Screen] or [Shift]-[Print Screen] to send the display to the printer.
- Printers you can specify include:

COLOR1	An IBM Color Printer with a black ribbon.
COLOR4	An IBM Color Printer with an RGB (red, green, blue, and black) ribbon.
COLOR8	An IBM Color Printer with a CMY (cyan, magenta, yellow, and black) ribbon.
COMPACT	A Compact Printer
GRAPHICS	The default printer (the one the system uses when you do not specify a printer) is an IBM Graphics Printer.
THERMAL	An IBM Convertible Printer.

- Switches work as follows:

/R	Prints colors as they appear on the display. Without the option, black prints as white and white prints as black.
/B	Also prints the background color. Without the option, the background color is not printed. The option works only with COLOR4 and COLOR8 printers.
/P=port	Specifies the parallel printer port. The default value is 1. Other values include 2 and 3.
/LCD	Prints from the Liquid Crystal Display (LCD) on the IBM portable computer.

- Not using the */R* switch, in essence, prints an inverse image of the screen.

- In text mode, it takes about 30 seconds to dump a screen. In graphics mode, it takes up to 3 minutes to dump a screen.
- In 320 by 200 color graphics mode with the COLOR1 or COLOR8 printer types, the screen prints in up to 4 shades of gray. In 640 by 200 graphics mode, the screen prints sideways on the printer. Other graphics modes print normally.
- You must restart DOS to deactivate GRAPHICS. In addition, each GRAPHICS command increases the size of DOS by about 3.3 Kbytes. Running the command many times can run you out of memory.

This page has no reference material.

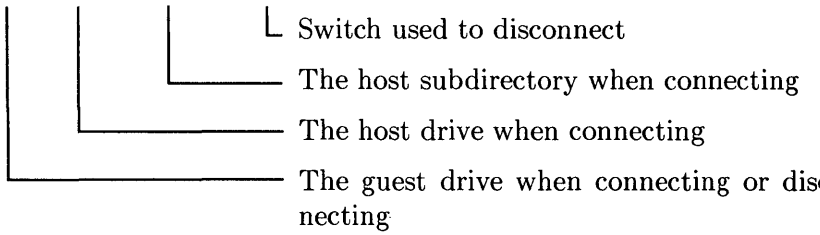
JOIN

External, Version 3.1 to present

Produces a single directory structure by connecting a disk drive to an empty subdirectory in the root directory of a second disk drive.

Syntax

`JOIN` [*d1:*] [*d2:\directory*] [*/D*]



Examples

Command	Result/Action
`join a: c:\projects`	Joins drive A to the subdirectory named *projects* in the root directory on drive C. (This is a typical example.)
`join a: /d`	Disconnects drive A from the host directory.

`c:\join`	Shows the current structure. For example:
	`A: => C:\projects`
	`B: => C:\data`

Tips and Hints

- The major uses of JOIN are to:
  1. connect two hard disks, which makes DOS think you have a very large hard disk; or
  2. connect a virtual (RAM) disk to a "real" disk.

  You can join other combinations of disks, but this is seldom necessary. In general, joining two disks can make other aspects of using DOS more difficult (for example, backing up files).
- Joining two drives makes the guest drive appear to be a part of the subdirectory on the host drive. You can access the guest drive only through the host subdirectory. Not using this path causes an error.
- You must specify a guest disk drive name (the *d1:*). This drive must not be the current drive.
- Not giving a host disk drive uses the current disk drive.
- You must specify a host subdirectory that is in the root directory. The host subdirectory can be the current directory. If the subdirectory does not exist, JOIN creates it. If the subdirectory exists, it must be empty.
- The guest drives must not be affected by ASSIGN or SUBST commands.
- You should not use the BACKUP, RESTORE, FORMAT, DISKCOPY, or DISKCOMP commands when two drives are joined. The DIR command works, but reports bytes free only for the host disk.
- While two drives are joined, CHKDSK processes the host drive, leaving the guest drive alone.

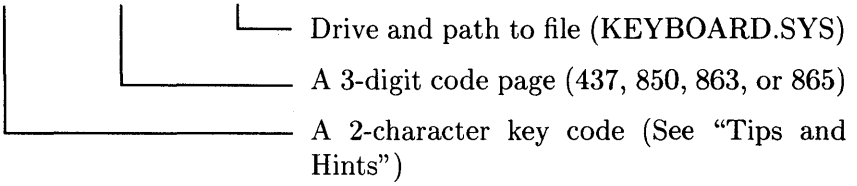
KEYB

External, Version 1 to present

Changes DOS so it supports a non-United States keyboard and codepages. (For this command, see the "Tips and Hints" section before you do anything. This should help you avoid errors.)

Syntax

KEYB [*keycode, codepage, filespec*]



Examples

Command	Result/Action
`keyb CF, 863, c:\KEYBOARD.SYS`	Changes the handler so it supports the Canadian keyboard and code page.
`keyb`	Displays the current keyboard and code pages.

Tips and Hints

- The key code values include:

Country	Code	Country	Code
Australia	US	Netherlands	DT
Belgium	BE	Norway	NO
English Canada	CE	French Canada	CF
Denmark	DK	Sweden	SV
Finland	SU	French Swiss	SF
France	FR	German Swiss	SD
Germany	GR	United Kingdom	UK
Italy	IT	United States	US
Latin America	LS	Spain	SP

- A specified *codepage* must be compatible with *keycode*. Acceptable combinations of code pages for key codes include:

Codepage	Acceptable Keycodes
437	BE, FR, GR, IT, LS, DT, SP, SU, SV, UK, US
850	Any keyboard code
863	CF, CE
865	DK or NO

- Do not use KEYB from earlier versions of DOS with DOS 3.3.
- Not specifying *codepage* uses the default code page for your country (that is, the code page established by the COUNTRY directive in CONFIG.SYS or, if you did not use the COUNTRY directive, the DOS default code page). This possibly is not what you want. You can specify 850, which works with all keyboard codes.
- You should specify KEYBOARD.SYS. If you don't, DOS looks for the file in the root directory of the current disk. If you don't give the drive or path, DOS uses the current drive or path. Beyond this, DOS uses the file name and extension to search for the file.
- Executing KEYB without parameters displays the active keyboard and code pages.
- Depending on the country, KEYB reassigns some characters and adds additional characters. This can change the layout of your keyboard, which means you probably need to accommodate new typing habits.
- Once a new keyboard layout is active, you can switch back to an American-English layout by typing CTRL-Alt-f1. Later, type CTRL-Alt-f2 to get back to the originally active layout.

Exit Codes

0	KEYB ran successfully.
1	Invalid key code, code page, or another syntax error.
2	Bad or missing KEYCODE.SYS file.
3	KEYB could not create a keyboard table in memory.
4	KEYB could not communicate with CON (the console).
5	The specified code page was not prepared.
6	Either: the internal translation table for a specified code page was not found; or the values of *keycode* and *codepage* are incompatible.

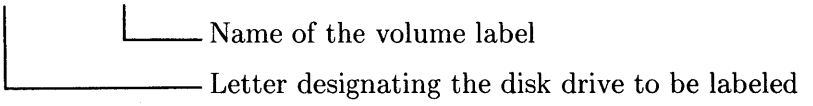
LABEL

External, Version 1 on

Creates, changes, or deletes a volume label for a disk.

Syntax

LABEL [*d: volume_label*]



Examples

Command	Action/Result
`label b:main_drive`	Makes *MAIN_DRIVE* the volume label for the disk in drive B.
LABEL	Since you did not specify a drive or volume label such as *B:MYDISK*), the command displays the volume for the current drive and asks if you want to enter a volume label. ■ Entering a volume label changes the existing label. ■ Just typing [Return] asks if you want to delete the current volume label. Entering *Y* removes the existing label; entering *N* exits LABEL, leaving the volume label intact.

Errors/Problems

You use a non-DOS disk	DOS reports an error and does not label the disk.
You enter an invalid volume label	DOS displays a warning and asks for a valid volume label.

Tips and Hints

- Specifying *dc, pathc, d:,* and *volume_label,* provides a sure means of creating a volume label with no interaction and without making assumptions about the state of the system.
- A volume label can be used by commands such as DIR, TREE, and CHKDSK.
- LABEL provides security when using FORMAT because the command requests a label to confirm your intent.
- Use 1 to 11 characters for *volume_label* as follows:
  - Upper- and lower-case letters and numbers 0 to 9
  - Space
  - The special characters $ # & @ ! -) _ (~ } ^ {
- Not specifying the drive, *d:,* ahead of *volume_label* changes the name of the disk in the current disk drive. Specifying the drive is safer.
- Do not label a SUBSTituted or ASSIGNed disk drive because DOS will label the *real* disk drive instead of the one you expect.
- Use the VOL command to see the existing label on a disk.

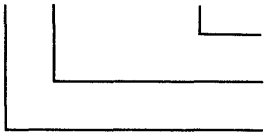
MKDIR or MD

Internal, Version 2 on

Creates a directory under the current or specified directory.

Syntax

`MKDIR [d:path]\directoryname`



- Name of directory you want to make
- Path to the directory
- Name of the disk drive containing the directory

Examples

Using MKDIR by Itself

Command	Action/Result
`mkdir c:\john\ltrs\personal`	Makes a directory named *personal* on drive C under the path named *\john\ltrs*.
`md junk.dir`	Makes a directory named *junk* on the current drive under the current directory. The directory has an extension named *dir*.

Using MKDIR With ASSIGN, JOIN, or SUBST

Command Sequence	Action/Result
`ASSIGN A = B` `MKDIR A:\tools\pie_cht`	Makes *pie_cht* under *\tools* on B (not on A because MKDIR uses the ASSIGNed disk).
`JOIN A: C:\old` `MKDIR A:\old\stuff`	Makes *stuff* under the root on A. During the JOIN, *stuff* is a directory on C under *\old*. When JOIN ends, *stuff* disappears from C but remains under the root on A. The SUBST command works similarly.

Tips and Hints

- The *directoryname* can have 1 to 8 characters (e.g. *jack, #123, ace-rpts*). You can append a 3-character extension (e.g. *new_doc.dir*). Do not have a space and avoid using ^ + / = \ ' : ? ; * [>] < . |
- Do not create a directory having the same name as an existing file (e.g. *reports* (a file) and *reports* (a directory)). You could have *reports.txt* (a file) and *reports* (a directory). You could also have *ltrs.txt* and *ltrs.dir*.
- A maximum *path* can have 63 characters including backslashes. Try to make an efficient and comprehensible set of directories.
- A backslash must separate *path* from *directoryname* (e.g. given *\users\bill* and *misc*, you have *\users\bill\misc*).
- Specifying *d:* but not *path* makes a directory under the current directory on the disk drive. Specifying *path* but not *d:* makes the directory under the path on the current disk drive.
- MKDIR can work with an ASSIGN, JOIN, or SUBST. Be careful. See the information about these commands before using them with MKDIR. You can easily create directories where you do not want them.
- Use RMDIR to remove directories; TREE to see existing directories, and APPEND or PATH to access other directories.

MODE (General Information and Errors)

The MODE command is eight commands in one. In general, the command sets the mode of operation for the printers, video display, and asynchronous communications adapter or it controls code page switching for the console and printer. You might want to include MODE commands in AUTOEXEC.BAT files depending on how you want applications to function.

Executing MODE

You typically execute MODE with a parameter such as LPT*x:*, COM*y:*, PREPARE, or SELECT. This means you need to know when to use MODE as-well-as how to use it. Besides using this reference material, read appropriate sections in the user's guide because some versions of MODE need to be executed in specific ways in a specific order.

Using a Variation of the MODE Command

1. To use a particular variation of the MODE command, examine the following list to find the variation you need. (You can skim the eight reference sections if you wish.)
2. Then, find the section having reference information for the variation.
3. Work through the section according to your need for information.

Finding Information About MODE

The reference information appears in sections because the MODE command is complex. This initial section discusses general information. Eight following sections discuss each variation as described in the following list. Study the list to see what MODE does, and then move to the corresponding section to find information.

- Set parallel printer characteristics (MODE LPT).
- Set display adapter characteristics (MODE display-type).
- Set asynchronous communications adapter features (MODE COM).
- Redirect printing from a parallel printer to a serial printer (MODE LPTx=COMy).
- Prepare a code page for use (MODE device CODEPAGE PREPARE).
- Select a code page for use (MODE device CODEPAGE SELECT).
- Reestablish the current code page (MODE device CODEPAGE REFRESH).
- Display code page status (MODE device CODEPAGE /STATUS).

MODE (Set Parallel Printer)

External, Version 1 to current

Sets IBM and Epson-compatible printer characteristics.

Syntax

`MODE LPT`*x:* [*cpl, lpi, P*]

- *P* — Continuous retries on time-out errors. (Ctrl-C overrides during printing.)
- *lpi* — Number of lines per inch (6 or 8).
- *cpl* — Number of characters per inch (80 or 132).
- *x:* — Printer number (1, 2, or 3). The colon is optional.

Examples

Command	Result/Action
`mode lpt3: 132, 6, P`	The *lpt3: 132, 8, P* uses the IBM-compatible printer on the 3rd parallel port, prints 132 characters per line at 8 lines per inch, and tries to print despite time-out errors (for example, jammed paper).
`mode lpt`	If you do not specify a value for *x* in *LPTx:*, MODE prompts for a parallel port number (1, 2, or 3) because this value must be given. To include options, type a colon (*:*), press the space bar, and type the options, separating each one with a comma (for example, *: 80, 6*). Without *P*, a time-out error makes the computer not retry printing.

Tips and Hints

- MODE LPTx: controls IBM Matrix and Graphics printers and all Epson-compatible printers. It might not work properly or at all on other printers.
- The characters-per-line and lines-per-inch options work only with IBM and Epson-compatible printers.
- The following items apply when using options:
  - You must provide a printer number for the parallel port.
  - Except for the printer number, an invalid option or skipped option has no effect on its existing state.
  - Besides the printer number, type a comma for an option you want to skip (for example, *: 80, , P* skips lines per inch).
- Concerning the *P* option for continuous retries:
  - Use CTRL-Break to abort a continuous retry.
  - To cancel continuous retries, execute MODE LPT without the *P*.
  - If you print to a deselected printer, the computer does not send a time-out error. It loops internally (appearing locked up) until the printer is selected (for example, turned on).
- Changing the column width sends printer-control code specifying normal font for 80 characters and condensed font for 132 characters.
- Changing the lines per inch sends printer-control code specifying 6 or 8 lines per inch and also sends printer-control code specifying the lines per page (66 lines for 6 lines per inch and 88 lines for 8 lines per inch).
- A MODE LPTx: cancels a MODE LPTx: = COMy:.

MODE (Set Display Adapter/Display Characteristics)

External, Version 2 to current

Switches the display adapter between the monochrome display and a graphics adapter/array on a two-display system. It also sets the characteristics of the graphics adapter/array. The adapter/array includes: a color/graphics adapter, an enhanced color/graphics adapter, or a video graphics array.

Syntax

`MODE` *dt*

or

`MODE` [*dt*,] s, [*T*]

- *dt* — Display type (40, 80, bw40, bw80, co40, co80, mono)
- s — Shifts the graphics display right or left (R or L)
- *T* — Aligns graphics screen with one-line test pattern

Examples

Command	Result/Action
`mode co80, r, t`	Sets the graphics display to 80 characters per line with color. Shifts the display to the right and provides a test pattern. (See section on Tips).
`mode bw40`	Sets the graphics display to 40 characters without color.
`mode mono`	Makes the monochrome display active.

Tips and Hints

- A monitor connected to an adapter or array is color or monochrome.
- When using *MODE dt, s, T,* you must specify *s* as *R* or *L*. When using *MODE dt,* you must specify the *dt* (display type) as follows:

40	Sets graphics display to 40 characters per line.
80	Sets graphics display to 80 characters per line.
bw40	Makes graphics display active, 40 characters, no color.
bw80	Makes graphics display active, 80 characters, no color.
co40	Makes graphics display active, 40 characters, color.
co80	Makes graphics display active, 80 characters, color.
mono	Makes the monochrome display active, no graphics.

- Not setting an appropriate display causes an Invalid parameter error.
- Setting color does not automatically produce color, but programs using color can produce a color display.
- Executing any valid form of the command clears the display.
- The *s* parameter works only with the color graphics adapter. You get an error for the monochrome adapter. In 40-column mode, the display shifts 1 character; in 80-column mode, it shifts 2 characters.
- The *T* option shows a test pattern with any graphics adapter (Convertible, Enhanced Graphics Adapter, Video Graphics Array).
- Using *s* and *T* shifts the display, shows the test pattern, and DOS asks if the display is aligned. Entering *N* repeats the shifting and prompting process. Entering *Y* returns to DOS.

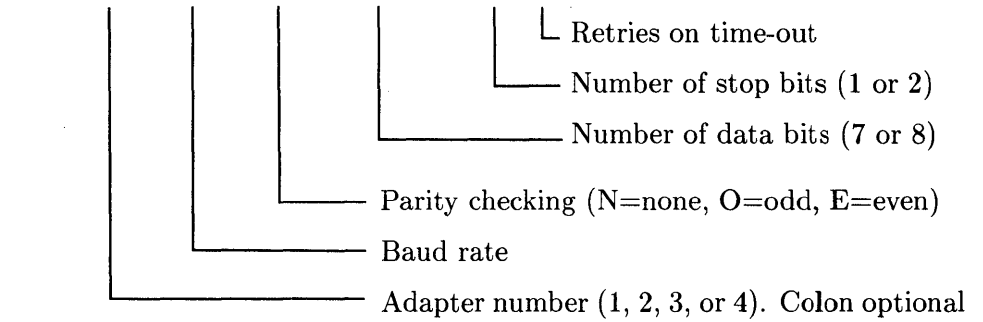
MODE (Set Asynchronous Communications)

External, Version 1 to current

Controls the protocol characteristics of a COM port.

Syntax

MODE COM*y*: *baud*[,*parity*,*databits*,*stopbits*,*P*]



Examples

Command	Result/Action
`mode com2: 1200, N, 7, 1`	The *com2: 1200, N, 7, 1* uses adapter number 2 at 1200 baud with no parity checking, 7 data bits, and 1 stop bit. Not specifying *P* makes the computer not retry on time-out errors.
`mode com`	The command prompts for the adapter number (1, 2, 3, or 4), which you must specify. You must provide a baud rate (typically 300, 1200, or 2400), but you do not need to provide additional options. To add options: type a colon, press the space bar, and include values for options you want, separating each with a comma.

Tips and Hints

- When using options, note the following items:
  - You must provide an adapter number (a value of 1, 2, 3, or 4 for *y* in COM*y:*), a space, **and** a baud rate (110, 150, 300, 600, 1200, 2400, 4800, 9600, or 19200. Check the device documentation. You can enter just the first two digits; for example, *24* for *2400*). (Do not use 19200 baud in the SoftPC environment.) If you type a colon, it must immediately follow the value for *y* with no space.
  - Except for the adapter number and baud rate, entering an invalid option has no effect, but you do get the message:

    ```
    Invalid parameter
    ```

    See the earlier section named "Mode (General Information and Errors)" see how to solve the problem.
  - If you do not want to change an option, enter a comma for its value (for example, `mode com4: 2400, , 7, 1, P` skips the parity option.
- When the adapter is set for continuous retries and a device is not ready, the computer appears to be locked up. Abort this loop by executing CTRL-Break.

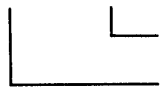
MODE (Redirect Printing: Parallel to Serial)

External, Version 2 to current

Makes DOS print to a serial printer instead of to a parallel printer. Printing normally goes to a parallel printer.

Syntax

MODE LPT*x:* [= COM*y:*]



y: is the async. comm. adpt. no. (1, 2, 3, or 4)

x: is the parallel printer number (1, 2, or 3)

(The colons are optional)

Examples

Command	Result/Action
`mode lpt1: = com4:`	Stops printing to the parallel printer at *1* and prints instead to the serial printer at *4*. To use the command, you first have to know the numbers you selected during configuration of DOS for the parallel and serial printers. (See "Tips and Hints".)

Tips and Hints

- Use this command when your system has only a serial printer or when your system has two printers and you want to print to the serial printer.
- You must use a valid number for the parallel and serial printer.
- After you execute the command, output from executing Shift-PrtSc goes to the serial printer.
- Use MODE COM*y:* to set up the serial adapter used for the serial printer before you use MODE LPT*x:* = COM*y:* to redirect printing.
- Cancel the command and return printing to the parallel printer by executing MODE LPT*x:* where *x:* is the parallel printer number you used in MODE LPT*x:* = COM*y:*.

MODE (Prepare Code Page for Use)

External, Version 3.3

Prepares (chooses) the code pages used with a device when you need to make DOS international. (This is a complex command; read the "Tips and Hints" before you do anything.)

Syntax

MODE *device* CODEPAGE PREPARE = ([*codepage, codepage, ...*] [*page_filespec*])

The parameters work as follows:

device — The device name for *codepage:*

CON:	The console.
PRN:	The first parallel printer.
LPTx:	The 1st, 2nd, or 3rd parallel printer where *x:* is 1, 2, or 3.

codepage — The number of the code page used with *device:*

437	United States
850	Multilingual
863	Canadian
865	Denmark and Norway

page_filespec — The disk drive having the code page information. The path to the file having the code page information if it is different from the path having MODE (for example, *\c_pages*). The name of the file having the code page information. Current code page files include the following:

4201.cpi	IBM ProPrinter
5202.cpi	IBM Quietwriter ™ III printer
ega.cpi	EGA or VGA displays

Examples

Study the following example and see the following paragraph for an explanation:

```
mode con codepage prepare = (850, 863) c:\ega.cpi
```

Prepares a code page for the console using multilingual and Canadian fonts. The command looks for EGA/VGA displays file,*ega.cpi,* in the root directory on drive C.

Here is a second example:

```
mode lpt1 cp prep = (,,437) c:\4201.cpi
```

The example shows how to replace one code page and retain previous code pages. The 437 code page replaces the code page held in the 3rd position; in this case, you have an IBM ProPrinter connected to the first parallel port using the IBM ProPrinter information file in the root directory for drive C.

Tips and Hints appear on the following page.

Tips and Hints

- Use MODE CODEPAGE PREPARE to prepare code pages that set the fonts for printers, the console keyboard, and the display. Execute the command before executing the MODE CODEPAGE SELECT subcommand.
- The IBM Quietwriter III printer is an exception because cartridges hold the font information. No MODE CODEPAGE PREPARE command is required when a cartridge has the required code page and the code page has been specified to the PRINTER.SYS driver.
- Memory holds prepared code pages in the specified order. Executing additional MODE CODEPAGE PREPARE commands replaces code pages only in the corresponding positions.
- You must specify *device,* CODEPAGE, and PREPARE. You can substitute CP and PREP, respectively.
- You must specify at least one code page. Separate additional code pages by commas and enclose the list in parentheses.

- You cannot have more code pages than the value of *added_codepage* specified to the appropriate device driver.
- When adding or replacing code pages, enter a comma for any previously specified codepage you do not want to change.
- Do not use hardware code pages (the page given to DISPLAY.SYS for the console and PRINTER.SYS for printers).
- Unless you specify paths for MODE and *pagefile.ext,* DOS uses the current drive and path.
- You must specify the name of the information file of the code page, *pagefile.ext.* You must use the extension if the file has one.
- Specifying an invalid code page number or and invalid file, removes all current code pages in the same position as the code pages specified in the command.

MODE (Select Code Page for Use)

External, Version 3.3

Activates a prepared code page used with a device.

Syntax

MODE *device* CODEPAGE SELECT = *codepage*

The parameters work as follows:

device — The device name for chosen codepage:

- CON: The console.
- PRN: The first parallel printer.
- LPTx: The 1st, 2nd, or 3rd parallel printer: *x:* is 1, 2, or 3.

codepage — The number of the code page for fonts used with the device:

- 437 United States
- 850 Multilingual
- 863 Canadian
- 865 Denmark and Norway

Examples

Command	Result/Action
`mode con codepage select = 863`	Selects a code page for the console using the Canadian font. The selected code page, *863*, must be part of a previously executed MODE CODEPAGE PREPARE command or be the hardware code page for the adapter/array.
`mode lpt1 cp sel = 850`	Activates the code page for the multilingual font for use with the printer attached to the first parallel port.

Tips and Hints

- You must specify *device*, CODEPAGE, and SELECT. You can substitute CP and SEL, respectively.
- The specified code page must be either:
    1. part of a MODE CODEPAGE PREPARE for a device; or
    2. the hardware code page specified to the device driver.
- MODE SELECT activates code pages for individual devices. Use the CHCP command to activate the code pages for available devices.
- MODE SELECT completes the downloading of any software font to a device. The IBM Quietwriter III printer is an exception because it uses cartridges. When you select a code page, the printer beeps. Make sure the appropriate cartridge is in the printer because DOS cannot check this. Using an inappropriate cartridge can print nonsense characters.
- MODE SELECT activates only a prepared code page or reactivates a hardware code page.

MODE (Reestablish the Current Code Page)

External, Version 3.3

Reloads and reactivates the code page used with a device.

Syntax

MODE *device* CODEPAGE REFRESH

device — The name of the device for reestablished code pages. (CON:, PRN:, or LPTx: x = 1, 2, or 3)

Examples

Command	Result/Action
`mode lpt3 codepage refresh`	Reestablishes the code page for fonts for the printer attached to the 3rd parallel printer port.

Tips and Hints

- You must specify *device*, CODEPAGE, and REFRESH. You can substitute CP and REF, respectively.
- MODE SELECT completes the downloading and reactivation of any software font to a device. Therefore, use the command when you turn a printer OFF and ON or after a program scrambles the display.
- If you don't give a code page, MODE REFRESH uses the last selected code page for the device.

MODE (Display Code Page Status)

External, Version 3.3

Displays the status of code pages for a device.

Syntax

MODE *device* CODEPAGE */STATUS*

A switch for displaying status.

The name of the device for reestablished code pages. (CON:, PRN:, or LPTx: x = 1, 2, or 3)

Examples

Command	Result/Action
`mode con codepage /status`	Displays the code page status for the console.
`mode lpt1 cp`	Displays the status of the printer attached to the 1st parallel port. Since you did not include /STATUS option, the command assumes you want the status to be displayed.

Tips and Hints

- You must specify **device** and CODEPAGE. /STATUS is optional. You can substitute CP and /STA, respectively.
- Including /STATUS in a command displays:
    1. the active code page, or shows that no code page is selected;
    2. the hardware code page, or pages;
    3. any prepared code pages; and
    4. available positions for additional prepared code pages.

  The displayed information is self-explanatory.
- If the status implies that you have a problem, use other MODE commands to solve it. For example, suppose the display shows this message:

```
No codepage has been SELECTED
```

  In this case, the device is using the hardware code page and will use this page until you use MODE with SELECT or you use CHCP to select a different code page.

MORE

External, Version 2 to present, Series 300 and 800

Displays output on your screen 24 lines at a time pausing if necessary. Input comes from the standard input device (usually the keyboard) or a redirected source (usually a file).

Syntax

```
MORE
```

Examples

Command	Result/Action
`more`	With no redirection to provide input, the cursor drops down on the display, waiting for you to type something. Typing echoes the characters on the display until you execute CTRL-Break. (This is not usually an effective way to use MORE.)
`more < letter`	The < redirects the input to MORE. This means that the file named *letter* provides the input to MORE; and MORE outputs the lines in *letter*, pausing when the screen is full.
`dir | sort | more`	This entry has two pipes, one between DIR and SORT, and another between SORT and MORE. 1. The output from DIR goes to SORT. 2. The output from SORT, a sorted list of files, goes to MORE. 3. MORE displays the sorted files.

Tips, Hints, and Details

- As a DOS filter, MORE accepts input and displays it on a screen one line at a time. When the output exceeds the size of the display, you see this message:

```
-- more --
```

  Pressing a key displays succeeding input, pausing whenever the screen fills up. This continues until all input to MORE is displayed. Then, you see this message:

```
Press any key to continue
```

- One screenful means 40 or 80 characters per line and 23 lines. The command wraps lines that exceed the current line length. When necessary, the command reserves two lines at the bottom of a screen to display a message.
- MORE works well when used in pipes or with redirection because you can manipulate a source of data and then see the filtered output a screenful at a time.

MOUSE

External HP extension to DOS, DOS Coprocessor only

Loads emulation support for a mouse. Once loaded, the standard Series 300 HP-HIL mouse emulates a Microsoft mouse in DOS applications.

Syntax

MOUSE [*/Cn /Sn /Hn /Vn /Dn*]

See "Tips and Hints" for descriptions of the options.

Examples

Command	Result/Action
`mouse /s30`	Loads emulation support for a mouse, and sets the horizontal and vertical sensitivity to 30 in a range from 0 to 100.

Tips and Hints

- Before you run a MOUSE command, you must assign the mouse to a COM port via the Device Configuration Menu (when you install DOS) or the *dos.cnf* file. The following entry shows how you could assign the mouse to COM2 by modifying the COM2 line in *dos.cnf:*

  ```
  COM2 MOUSE:/dev/locator
  ```

  Restart DOS after modifying the line.

- The following line removes the mouse driver from memory:

  ```
  mouse off
  ```

- The mouse emulation program increases the resident size of DOS and remains in memory until you restart DOS.

- Including a MOUSE command in your AUTOEXEC.BAT file loads support for the mouse each time you start up DOS.

- When you do not run a windowing system, DOS takes possession of the mouse when you start up DOS.
- When you run a windowing system, check the documentation for your system to determine how HP-UX and DOS share the mouse. You typically give possession of the mouse to DOS by moving the pointer to the border of the DOS window and pressing the right button on the mouse. On getting the popup menu, highlight the mouse option and press the right button again. To return the mouse to HP-UX, move the mouse pointer rapidly in a single direction, or type [Stop] followed by [M] to select the Release Mouse option in the DOS Coprocessor Main Menu.
- The following items describe how to use the options:

  /Cn Indicates the port currently assigned to the mouse. Use */C1* for *COM1,* and so on, up to 3. Using the option greatly reduces the time required to load support for the mouse.

  /Sn Sets the horizontal and vertical sensitivity of the mouse where *n* ranges from 0 to 100.

  /Hn Sets the horizontal sensitivity of the mouse where *n* ranges from 0 to 100.

  /Vn Sets the vertical sensitivity of the mouse where *n* ranges from 0 to 100.

  /Dn Sets the double-speed threshold, which determines the speed at which the cursor's motion is doubled on the screen. The value of *n* ranges from 0 to 100.

NLSFUNC

External, Version 3.3

Provides support in DOS for extended country information and lets you use CHCP (a command for changing code pages for many devices).

Syntax

NLSFUNC [*filespec*]

└ Drive, path, and file containing country information (COUNTRY.SYS in DOS 3.3)

Examples

Command	Result/Action
`nlsfunc`	Without a specification, NLSFUNC does not change the path DOS uses for look for a file containing extended country information.
`nlsfunc c:\bin\country.sys`	NLSFUNC finds *country.sys* on drive C in *bin* under the root directory.

Tips and Hints

- NLSFUNC is required only when your applications require the extended country information DOS can provide, or when you use CHCP to change code pages.
- The command loads the NLSFUNC program.
  - DOS increases in size (about 2.8 Kbytes).
  - NLSFUNC supports the DOS function call (66 in hex) that provides information for the current country code (set up by the COUNTRY code in your system or the COUNTRY directive in CONFIG.SYS).
  - You can use CHCP after loading NLSFUNC.
- NLSFUNC remains active until you restart DOS.
- It helps to designate appropriate drives and paths.
- If the extended country information file (usually COUNTRY.SYS) is not specified, NLSFUNC does not change the path DOS uses to find the file. The COUNTRY directive in CONFIG.SYS is the only other way to change the path. If neither NLSFUNC or CONFIG.SYS have set the path, DOS expects to use the COUNTRY.SYS file in the root directory.
- You can help avoid errors by placing COUNTRY.SYS in the root directory of the startup disk (probably drive C).

PAMCODE

External HP extension to DOS

Starts the Personal Application Manager command processor (PAM).

Syntax

```
PAMCODE ROOT [/P]
             └── Permanently loads the PAM command processor
```

Examples

Command	Result/Action
`pamcode root /p`	Installs a copy of the command processor named PAMCODE.COM. The */P* options tells DOS to load the the copy of PAMCODE.COM permanently.

Tips and Hints

- You can run PAMCODE from COMMAND.COM (the DOS command processor) or from PAM.
- Including */P* to load the PAM command processor permanently means the EXIT command will not return to the previous command processor.
- When you do not include */P*, running the EXIT command returns you to the previous command processor (typically COMMAND.COM).
- People most often run PAMCODE from the DOS command processor.

PATH

Internal, Version 2 to present

Sets the searching order for directories containing commands and batch files by telling DOS to search a specified list of drives and paths when a command or batch file is not found in the current directory.

Syntax

PATH [*d1:*]*path1*;[*d2:*]*path2*; ...

└── List of drives and paths containing DOS commands and executable files (do not include a filename or extension)

Examples

Command	Result/Action
`path a:\;c:\;c:bin;c:\bin\util`	Causes DOS to look for commands or batch files: 1. under the root directory on the disk in drive A; and 2. on the disk in drive C as follows: a. root directory. b. *bin* under the root directory. c. *util* under *bin* under the root directory.

Tips and Hints

- Use PATH to make DOS search your hierarchial file system, saving you the need to type drives and path names for commands and batch files.
- PATH applies only to commands, programs, and batch files having COM, EXE, or BAT extensions.
- The order of the specified drives and paths determines the search through directories for a command or batch file.
- Separate drive names and paths with colons and specifications with semicolons (for example, `c:\bin;a:\`).
- Not specifying a disk drive name for a path causes PATH to use the current disk drive.
- PATH might not always apply. Some applications expect files for data and libraries to be in your working directory.
- Specify absolute paths, not relative paths. That is, start paths at the root directory (for example, `c:\bin\util\local` instead of something like `util\local`).
- You can use APPEND to specify search path(s) for data files.

PRINT

External, Version 2 to present

Does background printing to a printer, which frees the computer to perform other tasks.

Syntax

$$\texttt{PRINT}\ \left[\begin{array}{l}/D{:}device\\ /B{:}bufsize\\ /M{:}maxtick\\ /Q{:}maxfiles\\ /S{:}timeslice\\ /U{:}busytick\end{array}\right]\ [filespec1]\ \left[\begin{array}{l}/P\\ /T\\ /C\end{array}\right]\ \ldots$$

In the syntax, you can specify additional files: *filespec2* and so on. See the section called "Tips and Hints" for information about the options.

Examples

Command	Result/Action
`print D:prn a:\rpts\memo.txt`	Selects PRN as the printing device. Prints the file named *memo.txt*, which is under the root directory on the disk in drive A.
`print letter.txt /P /C primer.doc`	Places *letter.txt* in the queue of files to be printed and cancels the background printing of *primer.doc*. (The order of the secondary switches, */P* and */C*, is important).

Tips and Hints

Concerning the options (switches) in the syntax, the first set of switches lets you determine how PRINT works. Specify what you want the first time you execute PRINT. Except for */D:device*, the switches manipulate how fast a printer works.

/D:device — The device used for printing. If used, it must be first, and if you do not use it, DOS prompts for a device. *PRN* works unless you have a serial printer. Then, enter the device name of the printer, typically *COM1*.

/B:bufsize — The size of the memory buffer used during printing ranges from 1 to 32767 (for example, *B:1024*). The default is 512. Use multiples of 512, such as 1024 and 2048. Using */B:4096* increases performance for most documents of two pages or less.

/M:maxtick — The maximum time in clock ticks that PRINT has for sending characters to a printer. The value ranges from 1 to 255 (for example, *M:4*). The default is 2, and it works well.

/Q:maxfiles — The maximum number of files you can queue for printing ranges from 1 to 32 (for example, *Q:20*). The default is 10. Placing too many files in the queue causes an error and causes DOS to ignore the command.

/S:timeslice — The number of slices in a second ranges from 1 to 255 (for example, *S:6*). The default is 8. It works well.

/U:busytick — The maximum time in clock ticks for PRINT to wait for a busy or unavailable printer. The value ranges from 1 to 255 (for example, *U:4*). Default is 1. Increase this time (about 2 to 8) as you use slower printers (for example, a 4800 baud serial printer or 100cps parallel printer).

More Tips and Hints for PRINT

Because you can use PRINT with many switches and file specifications, this section contains numerous tips and hints. The examples showed typical situations. If they do not provide enough information, study the following information:

- The term "*clock tick*" is the smallest unit of time for personal computers, approximately 0.0549 seconds.
- PRINT controls background printing (that is, printing a disk-based file while the computer performs other tasks). The command queues files to be printed (that is, it places them in line to be printed).
- PRINT works best with text files. Most characters, including control characters, get printed. DOS pads *CHR$(9)* with spaces to the next eight-column location.
- Using background printing for files having a *.COM* or *.EXE* extension can produce strange results.
- Do not print something else when PRINT controls the printer.
- Background printing of files on a flexible disk can take time. Do not remove a disk during printing.
- Do not edit or erase a file being printed.
- You can use a second set of switches when you use PRINT.
    - The order of the switches is important. Each switch affects the preceding file and succeeding files up to the next switch.
    - Not specifying a switch assumes */P*.

- The second set of switches follows:

  /P Turns on print mode, adding the preceding file and subsequent files to the print queue until finding a */C* in the command or until you type [Return].

  /T Terminates the background printing of all files, including the file being printed. You need not specify any filenames because the switch cancels printing of all files, including those on the command line.

  /C Cancels specific file(s) in the print queue. Stops a file being printed and displays a message. Removes the preceding file and all subsequent files from the print queue that were entered in a command until a */P* is found in the command or until you type [Return].

- PRINT is external. Therefore, you might need to provide a drive and path ahead of PRINT to help DOS find the command.
- Not entering a file name displays the status of the background printing.
- Once you set the status of background printing, you cannot change it until you restart DOS.
- Files print in the entered order. If you use wildcard characters, the files print in their order in the directory.
- Entering PRINT /C has no effect.
- If DOS detects a disk error during printing, DOS cancels printing, indicates there is an error, and continues by printing other files in the queue.
- The first execution of PRINT increases the size of DOS by about 5.2 Kbytes. Altering the default settings for PRINT can change the size of DOS. You must restart DOS to reclaim the lost space.

PROMPT

Internal, Version 2 to present

Lets you customize the DOS system prompt (for example, the A> or C> prompts).

Syntax

PROMPT *prompt_string*

└── The string (meta-string) for the prompt

Examples

Command	Result/Action
`prompt`	Makes the standard system prompt reappear. (Use this command to restore the system prompt.)
`Sue $`	Makes *Sue* the prompt and creates a space between the prompt and anything you type.
`prompt $p`	Makes the current disk drive and path the prompt (for example, *A:/ACME/JOBS/LTRS*). See meta-characters in "Tips and Hints".
`prompt $p $g`	Like the above example, but adds an => to imply where you see typed commands.
`prompt $d $$`	Makes the date and $ the prompt (for example, *Mon 8-22-1988* $). See meta-characters in "Tips and Hints".

Tips and Hints

- Meta-strings let you use special characters. Each meta-string starts with $ and has one additional character (for example, $*t*). Within practical limits, you can use several strings and meta-strings in a prompt.

$_	Does a CR LF sequence (moves the cursor to the next line).
$b	Produces a vertical bar (that is, $*b* displays |).
$d	Displays the date (for example, *Wed 7-20-1988*).
$e	Produces an Escape character, ASCII 1Bh.
$g	Displays the greater-than character (>).
$h	Does a backspace, which erases the previous character.
$l	Displays the less-than character (<).
$n	Shows the current disk drive in the prompt.
$p	Shows the current disk drive and path through the current directory (for example, *C:\DOS\BASIC*).
$q	Displays the equal character (=) .
$t	Shows the time in the prompt (for example, *9:54:12.14*).
$v	Shows the version number of your DOS.
other	Ignored (for example, $*k* does nothing).

- A new *prompt_string* becomes the system prompt until you restart DOS or execute another PROMPT. You can use PROMPT in *AUTOEXEC.BAT* files.
- To start a prompt with a DOS delimiter (blank, space, colon, and so on), precede the character with a null meta-string (for example, $*A*).
- Use SET to see the prompt string after setting it.

RECOVER

External, Version 2 to present

Recovers a single file or all files from a disc containing bad sectors. Data in bad sectors cannot be recovered. (Use RECOVER for one file with care; use RECOVER for all files on a disk with **extreme** care. See "Examples" and "Tips and Hints" before you use the command.)

Syntax

RECOVER *filespec*

└── The drive, path, and file name.

Examples

Command	Result/Action
`recover junk.doc`	Recovers the file named *junk.doc*.
`recover c:\report.txt`	Recovers the file named *report.txt* under the root directory on drive C.
`recover a:`	Removes all subdirectories and moves all files to the root directory, giving them meaningless names and appending trash to the end of all files not ending on a cluster boundary. on drive A. You probably never want to do this on your hard disk.

Tips and Hints

- RECOVER has serious limitations and can do things you may not like. Study the command and tips/hints before you do anything.
- If a sector on a disk causes read/write errors, RECOVER can be used to recover either the file containing the sector (minus the bad sector) or the entire disk if the bad sector is in the root directory.
- RECOVER asks about beginning recovery. If you decide not to continue, use [CTRL]-[Break] to stop execution. Otherwise, press any key to begin. DOS reads the file sector-by-sector, skipping over bad sectors, marking them as defective so DOS will not use them, and placing good sectors in a file having the specified name. After recovering good sectors, RECOVER displays:

  ```
  nnnnnn of mmmmmm bytes recovered
  ```

  where *nnnnnn* is the number of recovered bytes and *mmmmmm* is the size of the original file.

Tips and Hints continue on the next page.

- When you recover all files on a disk in a drive, DOS searches the File Allocation Table (FAT) for chains of allocation units (clusters). DOS creates a file in the root directory of the specified drive for each chain found, using this format:

```
FILEnnnn.REC
```

where *nnnn* is a sequential number, starting with *0001*. After recovering the files, RECOVER displays:

```
xxxx file(s) recovered
```

where *xxxx* is the number of recovered files. Noticing how the command can possibly move files and remove subdirectories, you can imagine potential problems. A DIR will display something like:

```
FILE0001 REC  5120 10-20-88 9:55a
FILE0002 REC  2048 10-20-88 9:55a
   .
   .
   .
```

To finish a recovery, examine the contents of the files, editing as necessary, and use RENAME to specify acceptable file names.

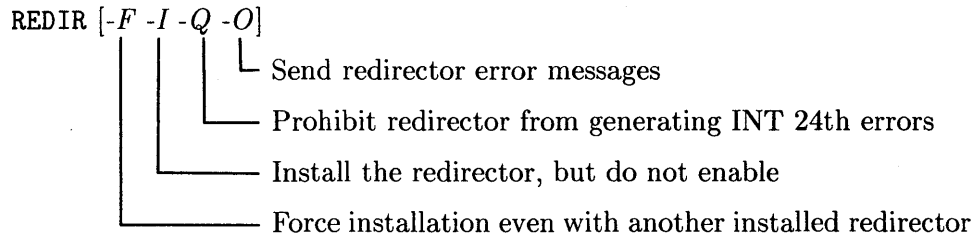
- If you have any doubt about where the command will do its work, provide appropriate drive and path specifications. Without specifications, the command works in the current drive and directory.
- When using a wildcard (* or ?), RECOVER acts only on the first file matching the wildcards.
- You cannot often use recovered files that are programs. You can typically use some of a recovered data or text file, but you may need to do some editing.
- When recovering a directory, you need to examine each file (named *FILEnnnn.REC* where *nnnn* is a four-place number). Rename the files as you determine their contents.
- Do not use RECOVER on a drive affected by an ASSIGN, JOIN, or SUBST command.
- RECOVER does not recover erased files.

REDIR

External HP extension, Version 3 to present, DOS Coprocessor Only

Installs HP-UX file system redirection into DOS. (Do this before using the DOSMOUNT command.)

Syntax



Examples

Command	Result/Action
`redir -f`	Forces installation of the redirector for DOS.

Tips and Hints

- You can use an upper- or lower-case letter for the switches.
- Running REDIR increases the size of DOS by 21890 bytes. The command remains in memory until you restart DOS.
- The command is integrated with DOS through the INT 2F vector, and relies on the semantics of DOS function calls.
- If you use the *-Q* switch, which prohibits the redirector from generating INT 24h errors, the redirector will fail the operation and return an error code to an application.
- You must run REDIR before using DOSMOUNT.

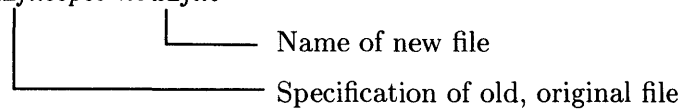
RENAME or REN

Internal, Version 1 to present

Changes the name of a file on a disk.

Syntax

RENAME *old_filespec new_file*



Examples

Command	Result/Action
`rename c:\notes.txt install.doc`	In the root directory on drive C, the command renames *notes.txt* to *install.doc*
`ren *.txt *.doc`	In the current drive and directory, renames all files having a *.txt* extension to files having a *.doc* extension. (There is a quirk in that RENAME would stop if it encounters a *junk.txt* and a *junk.doc* already exists.)

Tips and Hints

- You can easily make mistakes and rename the "wrong" file. It is better to carefully specify a drive and path than to assume you are in an appropriate drive and directory.
- You must specify an old (existing) file and a new (renamed) file.
- Since you rename an existing file, the new file has the same drive and path so DOS will know which file is being renamed. Do not specify a drive or path for the new file.

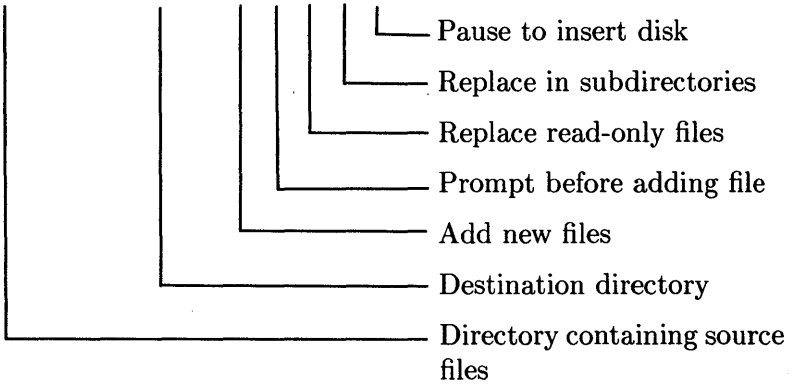
REPLACE

External, Version 3.2 to present

Selectively replaces files with matching names from one location to another, or selectively adds files from one location to another.

Syntax

REPLACE *source_filespec* [*d:path*] [*/A/P/R/S/W*]



Examples

Command	Result/Action
`replace a:\*.com c:\bin`	Copies all files having a *.COM* extension in the root directory on drive A **over those files having existing, matching file names** in the *BIN* directory under the root directory on drive C. Any *.COM* file not having a matching file in *C:\BIN* is not copied.
`replace a:*.exe c:\ /s`	Copies all files in the current directory of drive on drive A having an *.EXE* extension **over the existing, matching files** under the root directory on drive C and all subdirectories of the root directory.

Tips and Hints

- You must specify a filename. You can use wildcard characters.
- Not specifying a destination path replaces or adds files in the current directory.
- Do not use the */A* and */S* switches together.
- REPLACE replaces or adds files according to file names; the command does not care about the contents of files.
- Replace does not process hidden or system files.
- The following switches control how REPLACE works:

  /A Adds files to the destination directory. That is, you can add files from source that do not have a matching file name in the destination directory.

  /P Prompts for a decision about replacing or adding a file to the destination. You can replace or add files selectively, answering *Y* or *N* as desired.

  /R Replaces read-only files also. Without this switch, REPLACE skips read-only files.

  /S Replaces files in the destination directory and its subdirectories.

  /W Wait for insertion of the source disk. Helpful when you must change disks before REPLACE searches for source files. Without the switch, REPLACE assumes the correct disk is in the drive and begins the search for source files.

Tips and Hints continue on the next page.

- You generally use REPLACE to update DOS on a hard disk from a flexible disk. For the DOS Coprocessor, use the HP-UX *dosconfig* command to update a hard disk.
- Study what you want REPLACE to do before using it, especially in using */S*. Executing REPLACE with */S* on an entire disk can provide unexpected surprises because the command works on file names, not on the content of files.
- To prevent being surprised, carefully restrict the drives and paths of the source and destination files. It is better to replace files in small batches than to unleash a global action.

Exit Codes

REPLACE returns DOS error codes (numbers). Zero means files were added or replaced successfully. Otherwise, you can get the following codes:

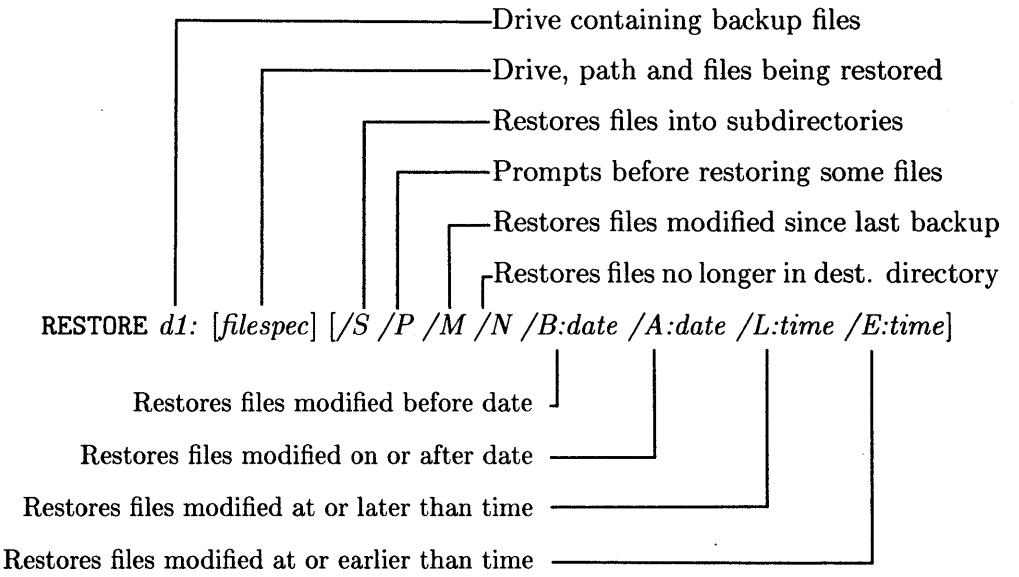
2 No source files found
3 Invalid source or target path
5 File or directory access denied
8 Out of memory
11 Invalid parameters (switches) or incorrect number of parameters.
15 Invalid disk drive
22 Incorrect version of DOS

RESTORE

External, Version 2 to present

Restores one or more backup files from a disk (flexible or hard) to another disk. (The complement of BACKUP.)

Syntax



Example

Command	Result/Action
`restore a: c:\util\*.*`	Restores all files (the *.*) in the *util* directory under the root directory on drive C, getting the files from the disk in drive A.

Tips and Hints

- Use the following switches to control a restore. Do not use switches */A*, */B*, and */N* in the same command.

  /S Restores files in the specified directory and its subdirectories.

  /P Prompts for decisions about restoring hidden or read-only files and files that were changed since the last backup. This switch works well when you use the */M* switch.

  /M Restores files that were modified or deleted since the last backup. A file is modified when its archive flag is on. Use this switch after having a disaster such as when the current copy of your modified files is bad, and you have erased files from your hard disk.

  /N Restores backed up files that no longer appear in the specified directory on the destination disk. To restore erased files use the */M* switch, and then use the */N* switch.

Tips and Hints continue on the next page.

/B:date Restores files created or modified *before* the date. Use the date according to its specification in CONFIG.SYS. This can be any of the following areas:

North American	*mm-dd-yy*
European	*dd-mm-yy*
East Asian	*yy-mm-dd*
mm	the month (1 through 12)
dd	the day (1 through 31)
yy	the year (80 through 99)

The character separating the parts of the date can vary: hyphen, period, or slash. In North America, you can use a slash instead of the hyphen.

/A:date Restores files created or modified on or *after* the date.

/L:time Restores files created or modified at or *later* than the time. Use the form *hh:mm:ss* for hours, minutes, and seconds. (for example, *02:30:15*). The separator is usually a colon, but you can use a period if the COUNTRY command in CONFIG.SYS was set to Denmark, Finland, or Germany. Do not use this switch without also using */A:date*.

/E:time Restores files created or modified *earlier* the time, using the same format as */L:time*. Do not use this switch without also using */B:date*.

- Only files saved with BACKUP can be restored.
- You must specify the drive containing the backed up files. If you put the restored files on the current disk, you need not specify a destination drive. You need to specify the path according to where you want to place the restored files. Not specifying a path puts the restored files in the current directory.
- Not specifying a file name is the same as specifying *.* (that is, all files are restored).
- If you use a path name, you must also specify a file name.

- RESTORE prompts you to insert a series of backup disks in order. Not inserting disks in the right order pauses the operation and you are asked to insert the correct disk.
- Restoring files backed up when an ASSIGN, SUBST, or JOIN command was in effect can produce unacceptable results. Clear these commands first.
- Use the */S* switch with caution. Version 3.3 of RESTORE does not restore DOS system files (*IBMBIO.COM, IBMDOS.COM,* and *COMMAND.COM*). In general, the 3.3 version of RESTORE is different from previous versions, so think about what you want to accomplish before specifying a command.
- Do not use the */M* switch while an *APPEND with /X* is in effect. APPEND makes RESTORE find files in paths specified to APPEND. Disable the APPEND. Then, do a RESTORE.

Exit Codes

The RESTORE command displays exit codes. Zero indicates a normal completion of a restore. Otherwise, you can get the following codes:

1 No files were found to restore
2 Some files were not restored due to conflicts in sharing
3 The restore was terminated by the operator via a CTRL-Break or an Esc
4 The restore was terminated by an error

RMDIR or RD

Internal, Version 2 to present

Removes the last subdirectory specified in a path.

Syntax

`RMDIR` [*d:*]*path*

- *path* — The path to the subdirectory to be removed
- *d:* — The drive containing the subdirectory

Examples

Command	Result/Action
`rmdir c:\util\tools\axe`	On drive C, the command removes the *axe* subdirectory, which is under *tools* under *util* under the root directory.
`rd bin\local\myprogs`	Starting in the current directory, the command starts relatively down in *bin* to *local* and then removes the directory named *myprogs*.
`rmdir temp`	Removes a subdirectory named *temp* in the current directory.

Tips and Hints

- The subdirectory must be empty except the current directory file (.) and any parent directory files (..).
- You cannot remove the current directory. The removed directory must be a relative subdirectory or a different directory given by an absolute path.
- You cannot remove the root directory.
- Not specifying a drive uses the current drive.

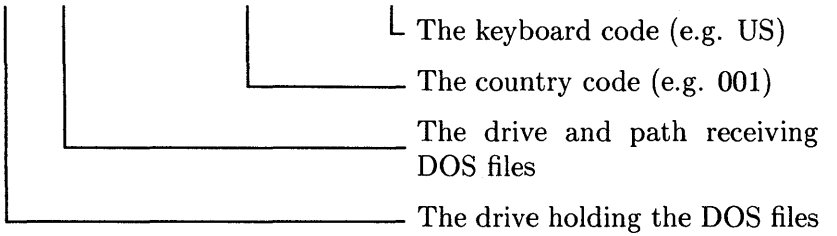
SELECT

External, Version 3 to present

Creates a disk with DOS files and prepares CONFIG.SYS and AUTOEXEC.BAT files configured for your country. (This is seldom needed (or easy to do) in the DOS Coprocessor environment.)

Syntax

SELECT [[*ds:*] *dd:*[*pathd*]] *countrycode keycode*



Examples

Command	Result/Action
`select a: c:\ 049 GR`	Gets the DOS files from the disk in drive A and places them in the root directory on drive C with the country code and keyboard code for Germany. Formats drive C before placing files on it.

Tips and Hints

- Specify *countrycode* and *keycode* according to the items in the table on the facing page:

Country	Country Code	Key Code
Austria	061	US
Belgium	032	BE
English Canada	001	CE
French Canada	002	CF
Denmark	045	DK
Finland	358	SU
France	033	FR
Germany	049	GR
Italy	039	IT
Latin America	003	LS
Netherlands	031	DT
Norway	047	NO
Spain	034	SP

- SELECT uses FORMAT to format the target disk (the disk that gets the files). Consequently, using SELECT on a disk having information destroys the information.
- Drive A is the default source drive, drive B is the default destination.
- The source drive must be drive A or drive B.
- Any drive except the source drive can be the destination drive.
- If the destination path does not specify a directory, SELECT copies files into the root directory.

SET

Internal, Version 2 to present

Adds, changes, displays, and deletes strings from the DOS environment.

Syntax

SET[*name*=[*string*]]

- *string* — The string to be assigned to name (e.g., $p)
- *name* — The name of the string in the environment

Examples

Command	Result/Action

`set`	With no parameters, SET displays the current environment; typically:
	`COMSPEC=C:\COMMAND.COM`
	`PROMPT=$p$g`
	`PATH=C:\UTIL;C:\BIN`

`set comspec=C:\apps\command.com`	Sets the string called COMSPEC to "C:\apps\command.com".

Tips and Hints

- Using SET by itself displays the current environment.
- Use SET *name=string* to set or change the string associated with a name.
- Use SET *name=* without the string to delete a name.
- Lower-case letters in *name* are changed to upper-case, and letters in *string* are taken literally.
- Using SET in an AUTOEXEC.BAT file for DOS can set the prompt and path. This way, you do not need to use the PROMPT or PATH commands to set those things.
- SET will set anything, including the meaningless name shown in the following example:

```
set junk=c:\trash
```

  The idea is to note requirements for applications and use SET in AUTOEXEC.BAT files to make the application work effectively. For example, some programs use environment variables (see their documentation). SET can set the values of these variables in the AUTOEXEC.BAT file for an application.
- An environment string should not exceed 121 characters. DOS enforces this by limiting the command line to 127 characters.
- DOS cannot expand the environment when you load resident programs such as MODE, GRAPHICS, KEYB, or PRINT. The default, initial size of the environment is 160 characters. The initial size can be increased to a maximum of 32768 characters by using the SHELL directive in CONFIG.SYS.

SIZE

External HP extension, DOS Coprocessor only

Displays the sizes of selected files in a subdirectory, and displays the amount of space needed to copy files to a disk.

Syntax

`SIZE` [*filespec*] [*d:*]

Will show space needed for a copy to the drive

Specifies directory for showing sizes of files

Examples

Command	Result/Action
`size d:\reports\old\*.txt`	Displays sizes in bytes of files in *\reports\old* having the *txt* extension.
`size c:\ a:`	Displays the amount of space in bytes required on drive A to hold all the files in the root directory on drive C.

Tips and Hints

- When you determine the sizes of files, the following example shows a typical listing:

```
size (c) Ace Software 1987
                report1.txt     3425
               myreport.txt      458
 ...
11 files, total of 125550 bytes
```

- When you determine the amount of disk space needed on drive A to contain the files in a specified directory, the following example shows a typical listing:

```
size (c) Ace Software 1987
              report1.txt     3425  ->   3072
             myreport.txt      458  ->   1024
 ...
11 files, total of 125550 bytes, (13096) bytes disk space required on A:
```

  Note that requirements appear in blocks of disk space.

- Running SIZE without parameters lists the sizes of all files in the current drive and directory.

- Running SIZE and specifying only a drive or directory reports the number of files and file sizes as 0. You must include a file name in the specification unless you are listing the files in the current directory.

- If you use SIZE to determine the required disk space for a copy, running DIR can tell you if the required amount of space is free on the destination disk.

SORT

External, Version 2 to present

Reads lines from the standard input device (typically the keyboard), does an ASCII sort of the lines, and writes the sorted lines to the standard output device (typically, the monitor).

Syntax

SORT [*/R /+c*]

- */+c* — Starts sorting at column number c
- */R* — Sorts in reverse order (Z to A)

Examples

Command	Result/Action
`sort <zip_code.txt`	Redirects input, getting the lines from the file named *zip_code.txt*. Displays output on the standard device, probably the display or a window on the monitor screen.
`sort <a:\names.txt /R | PRN`	Gets input through redirection from *names.txt* in the root directory on the disk in drive A. Sorts in reverse order and sends output via a pipe to the printer.
`dir | sort /+20`	SORT accepts input from the pipe from DIR and sorts the listing, starting with column 20. The remaining, sorted list goes to standard output.

Tips and Hints

- In general, use redirection and/or pipes. Without them, SORT typically takes input from the keyboard, which is not generally what you want to do. Without a pipe, SORT typically sends output to a display, which may or may not be what you want to do.
- If you redirect the input and output, use different respective file names.
- SORT uses an ASCII sequence as follows:
  - Lower-case letters are treated as upper-case letters.
  - Control characters, including Tab, are not expanded.
  - Characters in the 128 to 255 ASCII range can get collated according to rules set by the COUNTRY command.
- SORT can handle a file up to 63 Kbytes (64512 bytes).
- SORT works on text files, discarding information after and including the end-of-file character (EOF).
- SORT treats numbers as characters, 0 before 1, 1 before 2, and so on. Therefore, 142 or 1268 appear before 2 or 21, and so on.
- SORT does not expand control characters; consequently, a Control-I (Tab) may not be output as you expect.
- You can get unpredictable results when you sort files containing data from programs, compressed data, control characters, or unusual EOF characters.

SUBST

External, Version 3.1 to present

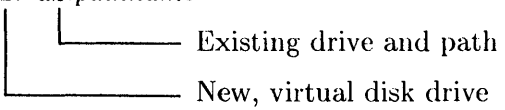
Substitutes a new drive, called a virtual drive, for an existing drive and path.

Syntax

The syntax varies according to your purpose.

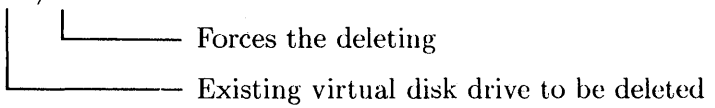
Establishing an Alias

SUBST *d1: d2:pathname*



Deleting an Alias

SUBST *d1: /D*



Seeing Current Aliases

SUBST

Examples

Command	Result/Action
`subst e: c:\bin`	Substitutes drive E for the subdirectory named *bin* under the root directory on drive C.
`subst e: /D`	Deletes the virtual drive E. The */D* switch causes the disconnect.

`subst`	Shows the current virtual drives, something like:
	`E: => C:\BIN`
	`F: => C:\USERS\DOCS`

Tips and Hints

- While a substitute drive is in effect, use caution in running DOS commands that work on directories or files. It is easy to affect the real entity instead of affecting the substituted entity.
- Do not use ASSIGN, JOIN, APPEND, CHKDSK, PATH, LABEL, BACKUP, DISKCOPY, DISKCOMP, FDISK, MKDIR, and RMDIR on a substituted drive.
- The virtual drive letter must be less than or equal to the LASTDRIVE letter defined in CONFIG.SYS, or it must be less than or equal to *F:* if LASTDRIVE is not set.
- *d1:* and *d2:* cannot be the same. *d1:* can be the same as the current drive.
- While a drive and path are substituted, you can still reference them by specifying the old drive and path.
- If you specify an existing physical drive as the virtual drive, *d1:*, that physical drive is inaccessible while the substitution is in effect.

SYS

External, Version 1 to present

Places a copy of hidden DOS system files on a specified disk.

Syntax

SYS [*d:*]

└ The name of the drive receiving the copy of DOS system files

Examples

Command	Result/Action
`sys a:`	Copies DOS system files onto the disk in drive A.

Tips and Hints

- You can use SYS to put DOS on application work disks. If you also copy *COMMAND.COM* onto the work disk, you can boot your system from the application disk.
- You must specify the drive name that gets a copy of DOS, and the current drive must contain files named *IBMBIO.COM* and *IBMDOS.COM*. Otherwise, SYS prompts you to insert a disk in the current drive having these files.
- To make a system disk a bootable disk, you must also copy *COMMAND.COM* onto the disk.
- A disk getting a copy of DOS must be completely empty, or must have been formatted with one of the following:
  - The */S* option (versions 1, 2, or 3).
  - The */B* option (versions 2 or 3).

  Alternatively, the disk could be formatted with a utility program that reserves space for the operating system. Otherwise, you get a message saying there is no room for the system.
- A system disk cannot start up DOS unless *IBMBIO.COM* and *IBMDOS.COM* are the first two files in the root directory (even though DIR does not list these files). In addition, *IBMBIO.COM* must reside in consecutive sectors. Not meeting these conditions causes an error.

TIME

External in version 1, Internal since

Sets or shows the system time.

Syntax

TIME [*hh:mm:ss.xx*]

- *xx* — Hundredths of a second (0 to 99)
- *ss* — Seconds (0 to 59)
- *mm* — Minutes (0 to 59)
- *hh* — Hour (0 to 23)

Examples

Command	Result/Action

`time`	Shows something like:
	`Current time is 13:23:54.23`
	`Enter new time`
	You can enter a time, which resets the time, or just type Return to continue.

`time 08:14:32`	Sets the hour to 8 a.m., the minutes to 14, the seconds to 32, and disregards the hundredths, which default to 0.

Tips and Hints

- The delimiters (colon and period) can vary, depending on the setting of the country code in CONFIG.SYS.
- You must enter the correct delimiters between numbers. The United States and similar countries use only the period to separate seconds from hundredths. Some countries use a comma.
- You do not need to provide all values. For example, *10:23* changes the hours and minutes and leaves the seconds alone.
- Not entering a valid time or delimiter causes an error.
- In the DOS Coprocessor and SOFTPC, time is set from the HP-UX time when you start up DOS. Changing DOS time does not change HP-UX time or the time on the battery-backed clock on your computer.
- The main reason for setting the time is to maintain information about when you create or change files and directories.
- When you start DOS without PAM or an AUTOEXEC.BAT file, DOS always prompts you to enter the time and date. If you have an AUTOEXEC.BAT file, DOS does not prompt for the time and date unless the file contains TIME and DATE commands.

TREE

External, Version 2 to present

Displays the directory structure on a disk and optionally displays the files in each directory.

Syntax

TREE [*d: /F*]

- *d:* — The disk drive having the directories you want to see
- */F* — Makes TREE also display the files

Examples

Command	Result/Action
`tree`	Displays a tree showing the directory structure of the current drive.
`tree c: /F >PRN`	Creates a tree of the directories on the disk in drive C, along with all the files in the directories, and outputs the results to the printer.

Tips and Hints

- The output from TREE can become cluttered. It is often better to redirect the output to a printer, and then study the tree in detail.
- TREE does not provide information about dates, times, and sizes.
- Getting a printed copy of the output from TREE is useful when you backup your system. Store the printout with the backup disks so you can see what the file system looked like.

TYPE

Internal, Version 1 to present

Displays the contents of a file on a display or in a window.

Syntax

TYPE *filespec*

└— The drive, path, and file to be displayed (The drive and path are optional)

Examples

Command	Result/Action
`type c:\util\tools\strip.scr`	Displays the lines in the script named *strip.scr*, which is in a directory named *tools* in a directory named *util* in the root directory on drive C.
`type junk | more`	Reads the lines in a file named *junk*, and the pipeline sends the output to MORE, which pauses the display each time the screen or window fills up.

Tips and Hints

- Not specifying a drive or path uses the current drive and directory.
- The specified file must match a file in the specified or current directory.
- You cannot use wildcard characters in the file specification.
- The command sends all characters in a file to the display, including control characters. The command expands a Tab character, Control-I, and CHR$(9) to the next 8-character column.
- TYPE does not work on a directory, only on a file.
- TYPE works well in pipelines and with redirection.
- If you want to pause the display when the number of lines in a file exceeds 23, pipe the output to MORE.

UX2DOS

An external HP extension to DOS

Converts ASCII files from the HP-UX format to the DOS format. (Besides working in DOS, the command works in HP-UX.)

Syntax

UX2DOS *filespec*

└— The file being converted

Examples

Command	Result/Action
`ux2dos autoexec.ux >autoexec.bat`	Converts an HP-UX file named *autoexec.ux* to the DOS format and redirects the converted output to a file named *autoexec.bat.*
`type config | ux2dos >c:.config.sys`	The TYPE command in the pipeline sends the file named *config* to UX2DOS for conversion, and the converted output from the UX2DOS command is redirected to a file named *config.sys* in the current directory on drive C.

Tips and Hints

- DOS cannot read HP-UX ASCII files directly. To use an HP-UX file in DOS, you need to first use UX2DOS to convert the file. For example, to use a word processor running on DOS to edit an HP-UX file, convert the file and then edit it.
- In most cases, UX2DOS works best with redirection and pipelines.
- Not specifying a file makes UX2DOS accept input from the standard input device, probably the keyboard.
- Unless redirected or piped, the converted output goes to the standard output device, probably the display.
- The original file that provides input is not changed.
- To get a more complete picture of converting DOS and HP-UX files, see the UX2DOS command.

VER

Internal, Version 2 to present

Syntax

VER

Examples

Command	Result/Action
`ver`	Could display something like the following example: `HP Vectra Personal Computer MS-DOS Version 3.30 - C.01.01`

Tips and Hints

- The command shows a one-digit version number and a two-digit revision number.

VERIFY

Internal, Version 2 to present

Turns the disk write verify switch off and on.

Syntax

VERIFY [*ON* | *OFF*]

- *OFF* — Turns the verify status off
- *ON* — Sets the verify status

Examples

Command	Result/Action
`verify`	With no options, you might see the following: `VERIFY is off` Otherwise, you will see it is ON.
`verify on`	Turns the verify status ON.

Tips and Hints

- VERIFY accepts only one of two options: *ON* or *OFF*. Without an option, you see the current status.
- The verify status remains on until you take one of the following actions:
  - Restart DOS.
  - Execute *VERIFY OFF*.
  - Do something that causes a *SET VERIFY* system call.
- VERIFY checks data written to a disk to ensure correct recording. When VERIFY is OFF, DOS does not check data. VERIFY does not affect any other DOS operations.
- Using VERIFY has a tradeoff:
  - Having the verify status ON ensures the integrity of data, but it takes longer to write a file to a disk.
  - Having the verify status OFF causes files to write faster, but you might have corrupted data. In most cases, you need not verify writing unless you must be sure the data is correct (for example, a company inventory) or you want to ensure a backup is correct.
- Having the verify status ON can slow up writing by up to 100%. The amount varies somewhat with the versions of DOS. Hard disks can take even more time.

VOL

Internal, Version 2 to present

Displays the volume label of a disk when the label exists.

Syntax

VOL [*d:*]

└The name of the disk drive

Examples

Command	Result/Action
`vol c:`	Displays the volume label of the disk in drive C. You might see something like this: `Volume in drive C is PROGRAMS`

Tips and Hints

- The command shows the volume label for the current drive if you do not specify a drive.
- You can create a volume label when using the FORMAT or LABEL commands.

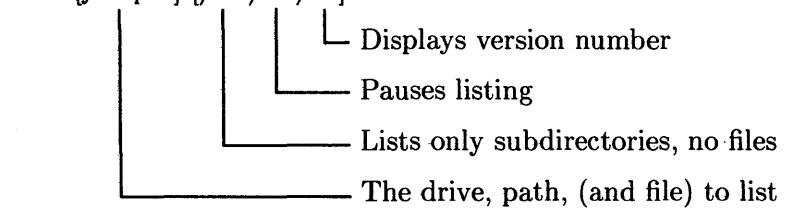
XDIR

External HP extension, Version 3.3, DOS Coprocessor only

Provides an HP-UX type listing of files.

Syntax

`XDIR` [*filespec*] [*/D /P /V*]



Examples

Command	Result/Action
`xdir d:\usr\lib\dos`	Lists the files and subdirectories in the *dos* directory under *d:\usr\lib*.

Tips and Hints

- XDIR is helpful for listing files on drive D, the drive having the HP-UX files.
- In particular, XDIR shows DOS file names, sizes in bytes, dates and times last modified, HP-UX file permissions, user/group ownerships, and HP-UX file names. For example, you might see the following:

```
..          <DIR>         7/02/88  2:38p [rwrwrw]  <115,016> ..
BIN         <DIR>        10/02/88  2:38p [rwrwrw]  <115,016> bin
LOAD_DOS             1876  8/11/88  4:06p [rwr-r-]  <115,016> load_dos
...
```

- Not specifying a drive or path uses the current drive and path.
- Specifying a file lists information only for that file. You can use wildcards to list a group of files. XDIR makes the following assumptions about specifications:

```
XDIR filename   means   XDIR filename.*
XDIR .ext       means   XDIR *.ext
```

- You can edit the COUNTRY directive in CONFIG.SYS to alter the format of the date and time.
- Files that do not meet naming conventions for DOS may not be accessible to DOS.
- At the end of a listing, XDIR displays the number of listed files and the amount of space remaining on the HP-UX file system. For example, you might see the following message:

```
12 File(s) 5564416 bytes free
```

3

Batch File Command Reference

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 executes the commands in the batch file, saving you from entering them yourself. Batch files contain the file name extension .BAT.

Batch processing has its own set of special commands which provide a rudimentary "program language", further enhancing the capabilities of MS-DOS batch processing.

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 batch files 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.

Batch Files

This section discusses batch files. A later section named "Batch File Commands" describes the commands you can use in a batch file.

Creating a Batch File

Use EDLIN, COPY or an ASCII word processor. The file must be unformatted, with each line containing one command line. When saving the file, name it with the .BAT extension, which indicates to MS-DOS that this is a batch file.

Executing a Batch File

Type the name of the file *without* the .BAT extension as shown in the following example:

```
runprog [Return]
```

The example executes a batch file named *runprog.bat* in the current directory of the active drive.

Preventing Command Line Echoing

Putting an @ character in front of a batch command prevents only that command from echoing to the display. This feature is commonly used with ECHO OFF, for example:

```
@ECHO OFF
```

Using Positional Parameters %0 - %9

You can insert ten positional parameters, %0 - %9, into batch command files to specify a replaceable parameter. The actual values for the replaceable parameters are entered on the command line when the batch command file is executed. The first value 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, without the .BAT extension. The %0 parameter is used in the following example for a batch file named *replace.bat:*

```
type %0.bat
dir %1
chkdsk %2
copy %1*.* %2
```

To execute this batch file, you must include two parameters in the command line to provide values for %1 and %2. The command line:

```
C>replace a: b:
```

substitutes a: for %1 and b: for %2. The command lines which will actually be executed are:

```
type replace.bat
dir a:
chkdsk b:
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 the command line contains fewer replacement values than the number of replaceable parameters in the batch file, the unspecified parameters will be deleted. For example, if the example above had been invoked with only one replacement value:

```
C>replace a:
```

the batch command file would execute the following MS-DOS command lines:

```
type replace.bat
dir a:
chkdsk
copy a:*.*
```

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 places name and parameter pairs in the system environment. To use a name in a batch file, enclose the name in percent signs (%). When the batch file is executed, the batch command processor will search the system environment for any names it encounters. The corresponding parameter will be substituted.

For example, the following command assigns the value *example.txt* to FILE:

```
A>set FILE=example.txt
```

To use this name in a batch file, enclose it in percent signs, as in this one-line batch file:

```
type %FILE%
```

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 execution.

Batch File Commands

This section describes the commands you can use in a batch file.

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 displays all command lines as they are executed. This is the default state.

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

ECHO followed by a message displays 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 demonstrates all four forms of ECHO.

```
ECHO OFF
ECHO This is a message displayed by ECHO
DIR A:/W
ECHO ON
ECHO
DIR A:/W
```

When executed, this file produces the following display:

```
C>ECHO OFF
This is a message displayed by ECHO

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>
```

Notes

1. To prevent the ECHO OFF command from echoing, put an @ character in front of it, for example:

```
@ECHO OFF
```

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:*

variable — 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 *set.*

set — 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)''
```

command — The command line executed at each iteration of the loop.

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 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
      :
      :
      :
```

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 to the line following a label.

Syntax

GOTO <*label*>

Operation

The label can be any string of characters. However, only the first eight characters are used by DOS. A label must be preceded by a colon (:) and there can 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 REM command following the label :TWO.

```
goto two
.
.
.
:two
rem label two
```

Notes

1. The GOTO command can be used with the IF command. 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

Allows conditional execution of MS-DOS commands in batch files.

Syntax

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

Operation

The IF command examines the *condition,* and executes the *command* if the condition is true. If NOT precedes the *condition,* the command line is executed when the condition is false. IF can test for the following conditions:

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:
	`IF %1==EXAMPLE`
	`IF %1==%2`

EXIST [*d:*] *[filespec]*	This condition is true if the file specified exists. The file name may include the wildcard characters.

Note

BACKUP, FORMAT, GRAFTABL, REPLACE, and RESTORE can return an ERRORLEVEL. This facility is provided for programs that may return an ERRORLEVEL that 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 file. (Pressing either of these key sequences will terminate the batch file.)

The PAUSE command is useful for issuing instructions to the 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 ...
```

Notes

1. If ECHO is OFF, the optional message will not be displayed. However, the `Strike a key when ready` prompt will be displayed.

REM

Purpose

The REM command displays a message on the screen.

Syntax

REM [<*message*>]

Operation

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:

```
Copy successfully completed.
```

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 to the replaceable parameters are shifted; %1 becomes %0, %2 becomes %1, and so on. The tenth value entered in the command line is assigned to %9.

With SHIFT, the limit of 10 replaceable parameters can effectively be removed. However, only 10 parameters can be active at any one time, and 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).

For example, suppose a batch file named *test.bat* contains four replaceable parameters %0 thru %3. The batch file is executed with five replacement values:

```
test value1 value2 value3 value4 value5
```

Thus, the replaceable parameters %0 thru %3 are initially assigned the following values:

%0 Value	%1 Value	%2 Value	%3 Value
test	value1	value2	value3

After the SHIFT command is executed, the values will be as follows:

%0 Value	%1 Value	%2 Value	%3 Value
value1	value2	value3	value4

The original value of %0 is lost, all values are shifted down one position, and the fourth replacement value given on the command line is used for %3. Another shift would use the final value on the command line, and produce the following values:

%0 Value	%1 Value	%2 Value	%3 Value
value2	value3	value4	value5

4

EDLIN Command Reference

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.

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.

To start EDLIN, enter:

```
EDLIN [path] <file name> [/B]
```

where:

path	Specifies the path to the file name you wish to create or edit. If omitted, the current directory and active drive are assumed.
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 enter EDLIN, you will see the following display:

```
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.

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 on the disc before you can load the remaining lines for editing. The WRITE and APPEND commands describe how to do this. To begin entering text, enter an I (Insert) command.

When you edit an existing file, EDLIN saves the old version of the file under the same file name but with the extension .BAK. 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 on the 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 on the disc.

Information Common To All EDLIN Commands

- Pathnames are acceptable as options to commands. For example, typing:

  ```
  edlin bin\user\joe\text.txt
  ```

  allows you to edit the file *text.txt* in the *bin\user\joe* subdirectory.

- 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 uppercase or lowercase 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.

  Mnewpage>

- 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 Return.

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
```

EDLIN Hints

- You can insert a control character (such as CTRL-C) into text by preceeding it with CTRL-V, then typing the letter (such as 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: *
```

- As with MS-DOS commands, EDLIN commands become effective only after you press Return.

EDLIN Syntax

The following syntax notation will be used in this chapter.

< >	Parameters enclosed by angle brackets represent data you must enter.
[ ]	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>]
```

The EDLIN parameters are:

1. <*line*> indicates you must specify a line number. Three possible entries can represent this parameter:

    a. 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.

    b. 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.

    c. 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
    ```

2. <*n*> indicates you must specify the number of lines. This parameter is used only with the Write and Append commands. Thus, when you use the <*n*> parameter, you enter the number of lines you wish to load from, or write to, the disc.

3. <*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 [Return] (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.

EDLIN Commands

EDLIN commands are described in detail in the remaining pages of this chapter.

The Append Command

Purpose

The Append command 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 load lines until memory is 75% full.

If your file is too large to load completely, you will have to edit that portion of the file that was loaded, then save some of the edited lines to the disc in order to free memory for more lines. You can then load the remaining lines into memory using the Append command. (Refer to the Write command for information on how to save edited lines to a 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. 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; the copied lines are 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
5:*HP Series 300 computers.
```

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
 5: HP Series 300 computers.
 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
10: HP Series 300 computers.
```

To copy lines and place them within this text, you must specify <*dline*> to tell EDLIN where to put them. 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
 5: HP Series 300 computers.
 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
10: HP Series 300 computers.
11: which is why Hewlett-Packard developed
12: HP Touch -- a special touch sensitive
13: screen that's available with
14: HP Series 300 computers.
```

In both examples, lines were 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 one or both are omitted. If you omit the first line, as in:

```
,<lline>d
```

deletion begins with the current line and ends with the line specified by <*lline*>. 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 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:

```
6d
```

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, which deletes from the current line (3) to line 6:

```
,6d
```

The result is:

```
1: This sample file has been written
2: to demonstrate dynamic line number generation.
```

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 [Return].

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 [Return], EDLIN will display the line; below the line to be edited, EDLIN will display the line number. If you only press [Return], 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 [Return]. 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 "Executing Commands" in *Using the Series 300 DOS Coprocessor*. The existing line serves as the template until the [Return] 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 [Return]. Use the arrow 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 arrow keys to position the cursor.

Example

Assume the following file 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 [Return], and see the following display:

```
*4
   4:*four.
   4:*
```

To edit the line using the MS-DOS editing keys, enter:

To Do This ...	Type This ...	EDLIN Response
Enter Insert mode	[Insert char]	
Type in the data	number	4: number_
Copy All	[f3]	4: number four.
Save changes	[Return]	*

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 [Return].

The End Command

Purpose

The End command ends the editing session, saves the edited file on the disc, and returns you to the MS-DOS command prompt.

Syntax

```
E
```

Operation

The End command saves the edited file on the disc under the name 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 current directory of the active drive unless you named a different drive and directory when you entered EDLIN.

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 the disc, if any, will have an 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 Return.

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 CTRL-Break or CTRL-Z to leave Insert mode and return to the EDLIN prompt.

Examples

Assume the following file is ready to edit:

```
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 line 4, 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-Z. 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.
```

Placing the two inserted lines at the end of file 1 gives:

```
 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
 6: (the D and I commands)
 7: to edit text
 8: in your file.
 9: First new line of text
10:*Second new line of text
```

Note placement of the * to mark the current line in each example.

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 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 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 is ready to edit:

```
 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:

```
2,51
```

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:

```
.,261
```

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:

```
1
```

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 destination line number 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 from (current line) to (current line + 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:

```
20,30,100m
```

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:

```
1p
```

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 want to exit EDLIN without saving changes made during the current editing session. The following message is displayed:

```
Abort edit (Y/N)?
```

Press [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.)

Press [N] to continue the editing session.

Caution When you start EDLIN and enter an existing file name, EDLIN erases the previous copy of the file with the extension of .BAK to make room for 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 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 string, 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], or until you press [Return] if <*string2*> is omitted.

<*string2*> begins immediately after [CTRL]-[Z] and continues until you press [Return].

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] [Return]

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 content 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:

```
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:*This is the last line.
```

To replace the string "and" with the string "or" in lines 2 through 12, enter:

```
2,12 Rand[CTRL]-[Z]or[Return]
```

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[Return]
```

will be executed in the following way:

```
4: Delete or Insert
O.K.? Y
5: (The D or I commands)
O.K.?Y
5: (The D or I commors)
O.K.?N
8: The insert commor can place new lines
O.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 Return.

Operation

The string you wish to have matched by the Search command must be exact as to uppercase and lowercase letters. For example, if you enter a search string in all uppercase letters, only uppercase occurrences will be matched.

When the first line containing the specified string has been found and displayed, the search will end unless you have used the ? option. 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 Return, the line will become the current line and the search will terminate. If you press any other alphanumeric key, 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 of the first line or last line parameters, the system provides default values. If you omit the first line parameter, <*line*> 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 this parameter.

EDLIN has exact 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 [Return].

You can place more than one command on a line containing a Search command by terminating the Search command with [CTRL]-[Z]; then begin the next command in the following character position.

Examples

Assume the following file is ready to edit:

```
1: This is a sample file that
2: will demonstrate
3: the Search 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:

```
1,7Sand
```

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 comm*and*. 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 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.
O.K.? N
     6: and required string
O.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 the file specified by <*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 the 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.

5

DEBUG Command Reference

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 advanced 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.

Entering 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:

```
dcs: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 uppercase and lowercase letters may be used in DEBUG commands and their parameters.

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	Action
<*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 Series 300 DOS Coprocessor" 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 "string" is legal'
'This ''string'' is legal"
"This ""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 Series 300 DOS Coprocessor

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:

```
dcs:100     (offset)
d20A2:100   (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 in the code segment register, and
100H	is the offset in hexadecimal.

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. 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 computer 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. Exit the current DOS session, then start a new one.

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 *<filespec>* 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* (an optional manual you may order) 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 FFFEH.

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 Commands

The remaining pages in this chapter describe DEBUG commands in detail. 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     WO [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 Return 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:

```
dcs:0100 L 20
```

or

```
cs:0100 011F
```

DEBUG displays the dump in the following format:

```
207E:0100  48 50 20 56 65 63 74 72-61 20 54 6F 75 63 68 20
207E:0110  53 63 72 65 65 6E 00 00-00 00 00 00 00 00 00 00
```

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 Return to terminate the Enter command. You can press Return 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 SPACE bar 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 [Return] 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 MS- DOS 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 MS-DOS function calls can be used inside 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 0F 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:

```
02F8 4F
```

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

Because MS-DOS function calls can be used inside 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 (:) prompt. You can then either enter a <*value*> to change the register, or you can press [Return] if no change is needed.

Valid register-names are:

```
AX      BP      SS
BX      SI      CS
CX      DI      IP
DX      DS      PC
SP      ES      F
```

Note IP and PC both refer to the Instruction 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 Return 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 [Return]
```

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 [Return] 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	Bad flag 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.
BP	Too many breakpoints You specified more than ten breakpoints as parameters to the G (Go) command. Retype the Go command with ten or fewer breakpoints.
BR	Bad register You typed the R command with an invalid register name. See the Register command for the list of valid register names.
DF	Double flag You typed two values for one flag. You may specify a flag value only once per RF command.

6

MS-LINK Reference

This chapter describes the MS-LINK utility. The first part of this chapter describes MS-LINK 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 50 Kbytes of memory (40 Kbytes for code and data, and 10 Kbytes 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 links 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).

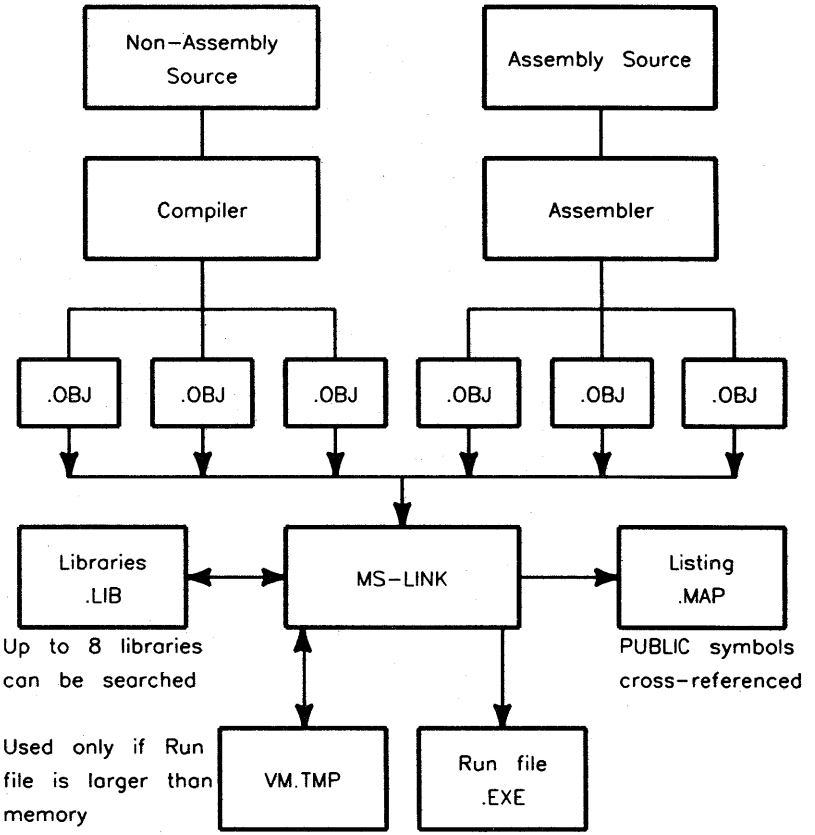
As MS-LINK combines the object modules into an .EXE file, it resolves any external references that the object modules make to symbols that are defined in other modules. MS-LINK can then search one or more library files to find the definition of any external references that it has not been able to resolve.

MS-LINK also produces a list file (.MAP) 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.

The following diagram shows the relationship between MS-LINK and other elements of the system.



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 64 Kbytes 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 64 Kbytes 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 64 Kbytes of memory. If they don't fit, MS-LINK displays an error message saying so.

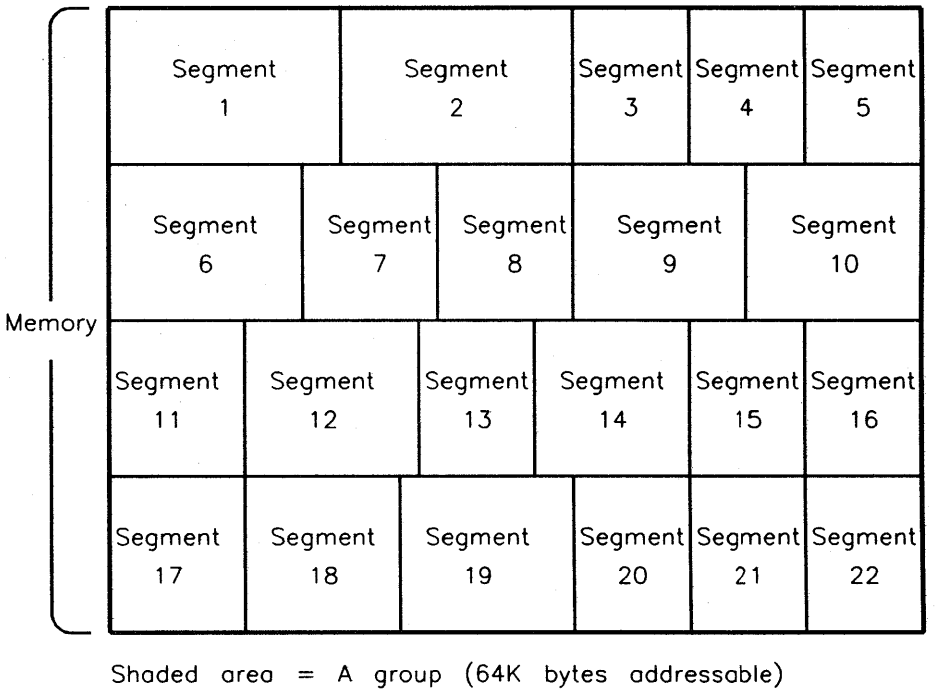
A **class** is a collection of segments. The placement of segments in a class controls the order and relative placement of segments in memory. The segments in a class need not have the same address. The user assigns the class name in an assembly-language program. For high-level languages (such as BASIC, 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 64 Kbyte 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 64 Kbytes long,
- a **group** is a set of segments that occupies up to 64 Kbytes,
- and a **class** is a collection of contiguous segments that can be loaded into memory consecutively across 64 Kbyte boundaries.

The following diagram illustrates the relationship between segments and groups in memory.



How MS-LINK Combines and Arranges Segments

MS-LINK can join the following modules of object code.

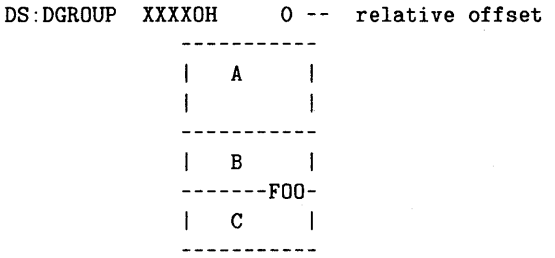
- Private combinations
- Public combinations
- Stack combinations
- Common combinations

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.

The following items summarize 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.)

Type	Combination Procedure
Private	Segments load separately, and stay separate. They can be contiguous physically but not logically, even segments having the same name. Each private segment has a base address.
Public	Segments having the same segment and class names load 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 types are treated like public types, except 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.

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.



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 64 Kbytes). 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, they will be loaded in the indicated order:

```
A SEGMENT 'FOO'                FOO

B SEGMENT 'BAZ'                  A

C SEGMENT 'BAZ'                  E

D SEGMENT 'ZOO'                BAZ

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

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

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 following four types of files.

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 prompt does not require that you specify the filename extension.

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

```
E:\BASIC\FNTOUCH.OBJ'
```

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 following extensions.

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 below.

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 below.

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 "Command Prompts" for more information.

The following items list the default extensions that MS-LINK assumes if you do not specify filename extensions in the first four items in the command line.

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 Return), MS-LINK will display the prompt for which a response is missing, and will wait for you to enter a valid response from the keyboard. Once you have done so, MS-LINK continues the linking session.

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 [Return] 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
COBLIB.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 [Return] 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, `C:\>`) 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.)

The following items show the four prompts and the corresponding actions that you can take.

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-Object- filename.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
00DB0H    00FAFH    0200H     STACK     STACK
00FB0H    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 [Return] 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 (available with and documented in the Programmer's Tools); or created as part of a pre-packaged library, such as the library available with a high- level 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, after inserting the disc with the library file on it.

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 Return at the end of the line. If the Plus-sign Return 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 Return—not with a Plus-sign Return.

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+Return
Object Modules [.OBJ]: HAPPY+FRODO+Return
Object Modules [.OBJ]: DOC Return
```

The Semicolon

Any time after you respond to the `Object Modules` prompt, you can use a single semicolon (;) followed immediately by [Return] to select the default responses to the remaining prompts. Entering this command saves time and avoids the need to enter a series of [Return] 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 [Return] keystrokes to do so.

For example, consider the prompts and responses shown below:

```
Object Modules [.OBJ]: SLEEPY GRUMPY [Return]
Run Module [SLEEPY.EXE]: ; [Return]
```

Because a semicolon appears directly before the [Return] 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).

CTRL-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.

Note that [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

The following items summarize the 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 [Return] 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 64 Kbytes, 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 message:

```
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. The next three tables show a typical linker map.

Start	Stop	Length	Name
00000H	009ECH	09EDH	CODE
009F0H	01166H	0777H	SYSINITSEG

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

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 [Return] 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
```

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, and to pass over the Library search.

```
B>link
Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983, 1984, 1985
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, 1984, 1985

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, 1984, 1985

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 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>
```

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, 1984, 1985

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, 1984, 1985

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, 1984, 1985

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:

- 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 LINKOBJS and used the MS-DOS TYPE command to view the contents. The command to call and run MS-LINK is followed by

```
LINK @LINKOBJS
```

enabling the response file to respond to the four prompts.

```
B>TYPE LINKOBJS
MAIN+SECOND+BIGFILE+
E:\BASIC\FNTOUCH+C:CHAR ;

B>LINK @LINKOBJS

Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983, 1984, 1985

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

The response file method is again used. The Linker was unable to locate the Object Module LONGFILENAME.OBJ and prompted to 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, 1984, 1985

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, 1984, 1985

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, 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, 1984, 1985

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  STACK_SE  STACK

Origin   Group
F0EE: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, 1984, 1985

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 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
```

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, 1984, 1985

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, 1984, 1985

B>A:
A>SAMPLE Simple linker example

A>B:

B>
```

Example 12

This example demonstrates use of the /PAUSE Switch.

```
B>LINK SAMPLE/P;

Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983, 1984, 1985

About to generate .EXE file
Change disks <hit ENTER> [Return]

B>
```

Example 13

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, 1984, 1985

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, 1984, 1985

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>
```

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 virtual-memory 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: PROGRAM SIZE OR NUMBER OF SEGMENTS EXCEEDS CAPACITY OF LINKER

Cause: The aggregate size of all of the files, once linked, cannot exceed 384 Kbytes; 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 64 Kbytes was requested.

Remedy: Use the /STACK switch to specify a stack size that is equal to or less than 64 Kbytes.

MESSAGE: SEGMENT SIZE EXCEEDS 64K

Cause: 64 Kbytes is the upper limits of the addressing system.

Remedy: Reduce segment size by splitting it into two or more segments.

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: SYMBOL TABLE CAPACITY EXCEEDED

Cause: Many long names were entered (occupying approximately 25 Kbytes).

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: 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.

A

DOS Message Directory

MS-DOS messages (called DOS hereafter) are issued when DOS or a DOS command encounters an error, needs to inform the user of system status, or prompt the user for an action.

A DOS command or the resident portion of DOS can generate an error message. The resident portion generates a message only when a disk or device encounters an error condition.

The following section named "Disk and Device Errors" discusses messages the resident portion can generate. A final section named "DOS Command Messages" lists messages generated by commands in alphabetical order. After each message, you see a table that has the following parts:

Command:	Lists the command, or commands, that can generate the error message.
Cause:	Describes the condition or conditions in the system that causes the error.
Action:	Explains the actions you take to remove the error condition, or notes that no action is required.

Disc and Device Errors

If a disk or device error occurs at any time during a command or program, DOS displays an error message in the following format:

```
<type><action> drive <x>
Abort,Ignore,Retry:
```

The error conditions for <type> can be one of the following:

```
Bad call format error
Bad command error
Bad unit error
Data error
FCB unavailable
General failure
Lock violation
Invalid disk change
No paper error
Non-DOS disk error
Not ready error
Read fault error
Sector not found error
Seek error
Sharing violation
Write fault error
Write protect error
```

The <action> can be reading or writing. The drive designator indicates the drive in which the error occurred. For example, attempting to write to a disk in drive A that had a write-protect tab produces the following error message:

```
Write protect error writing drive A
Abort, Retry, Ignore?
```

DOS waits for you to respond in one of the following ways:

A - Abort	End the program requesting the disk read or write.
I - Ignore	Ignore the error condition and proceed with the program.
R - Retry	Repeat the operation. Use this response when you have corrected the error.

Besides the error conditions mentioned above for disks and devices, the resident portion can display the following error message:

```
FILE ALLOCATION TABLE BAD FOR DRIVE x
```

The copy in memory of one of the allocation tables has error. Run CHKDSK with the */F* option, and if the error persists, the disk is unusable. Reformat it before using it again.

DOS Command Messages

This section contains the error messages a DOS command can display. Look for your message in the alphabetical list. The table following the message describes the situation.

Abort Edit (Y/N)?

Command:	EDLIN
Cause:	You entered a Q session without saving any editing changes.
Action:	Enter Y to exit without saving the editing changes to disk. Enter N to cancel the command and continue the editing session. Use E to exit EDLIN and save editing changes.

Abort, Retry, Ignore?

Command: DOS

Cause: A disk or disk device error has been detected by DOS during a program or command's execution.

Action: Enter A to abort the program, R to have DOS attempt the operation again, or I to proceed with the program or command. The I option is not recommended.

ABORTED - Assigned Port Number missing or invalid

Command: MODE

Cause: You specified printer redirection and the specified serial port is not present in the system, or you specified a port other than COM1 or COM2.

Action: Re-enter command line.

ABORTED - Bad device parameter keyword

Command: MODE

Cause: You specified a device other than LPTn or COMn. MODE only affects the parallel ports, serial ports, and screen.

Action: Re-enter command line.

ABORTED - Illegal BUSY value

Command: MODE

Cause: You specified something other than P to initialize a port as the printer device.

Action: Re-enter command line.

ABORTED - Illegal STOPBITS value

Command: MODE

Cause: You specified a value other than 1 or 2 for the number of stop bits.

Action: Re-enter command line.

ABORTED - Illegal DATABITS value

Command: MODE

Cause: You specified a value other than 7 or 8 for the number of data bits.

Action: Re-enter command line.

ABORTED - Invalid Switch

Command: MODE

Cause: An invalid parameter was given in the command line.

Action: Re-enter command line.

Active code page not available from CON device

Command: KEYB

Cause: There is no currently loaded CON code page, or the DEVICE=DISPLAY.SYS directive was not given in CONFIG.SYS.

Action: For CON, prepare and select a code page, or set the directive in CONFIG.SYS. Then, restart DOS.

Allocation error for file, size adjusted

Command: CHKDSK

Cause: The file has an invalid sector number. The problem exists in the FAT, not in the file. CHKDSK truncates the file at the end of the last valid sector.

Action: If rerunning CHKDSK does not fix the problem, you can restore the backup copy of the file (be aware that RESTORE has some limitations).

Allocation error, size adjusted

Command: CHKDSK

Cause: The size of the indicated file is inconsistent with the amount of data allocated to the file.

Action: If you specified /F, the file is truncated at the end of the last valid cluster. If you did not specify /F, the message is just information. CHKDSK takes no action, so you should re-run the command with /F.

Are you sure (Y/N)?

Command: DEL, ERASE

Cause: Appears when you instruct DOS to delete all of the files in a directory.

Action: Enter Y to proceed, N to abort the command.

APPEND/ASSIGN Conflict

Command: APPEND

Cause: You tried to use APPEND after running ASSIGN.

Action: Cancel the ASSIGN, execute APPEND, and then execute ASSIGN again.

Bad command or file name

Command: DOS

Cause: DOS is unable to find the program or command just entered.

Action: 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 DOS where to look for the command.

Bad file number

Command: FC

Cause: FC finds a defect in one of the files specified.

Action: Run CHKDSK to verify the integrity of your disk.

Bad or missing (filename)

Command: DOS

Cause: 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 disk, or an error is detected in the driver as its being loaded. The name of driver (filename) is displayed in the error message. In either case, the device driver is not incorporated into DOS.

Action: Check the spelling of the filename in CONFIG.SYS or copy the file containing the device driver into the root directory of the boot disk.

Bad or missing Command Interpreter

Command: DOS

Cause: DOS cannot 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 disk or if an error was encountered as the file was being loaded.

Action: Copy the file containing the command processor into the root directory of the boot disk or check the spelling of the SHELL filename in the CONFIG.SYS file.

Bad or missing Keyboard Definition File

Command: KEYB

Cause: KEYBOARD.SYS is corrupted, or KEYB could not find the file.

Action: Try again, making sure you specify the drive and path to the file. If you get the message again, the file is corrupt. Fix the file according to directions given in your user's guide for editing KEYBOARD.SYS.

Buffer size adjusted

Command: VDISK (a directive in CONFIG.SYS)

Cause: It is necessary to adjust the buffer size specified in the DEVICE=VDISK.SYS command in the CONFIG.SYS file.

Action: Edit CONFIG.SYS as required to make the buffer work.

Canceled by operator

Command: PRINT

Cause: You specified the /T switch in the PRINT command.

Action: Try the command again without the switch.

Cannot CHDIR to (path) - tree past this point not processed

Command: CHKDSK

Cause: The command cannot reach the specified subdirectory. All subdirectories beneath this directory will not be verified.

Action: No action required.

Cannot CHDIR to root. Processing cannot continue

Command: CHKDSK

Cause: The command cannot return to the root directory while it is verifying the tree directory structure. CHKDSK cannot continue checking the subdirectories in the root.

Action: No action required.

Cannot CHKDSK a Network drive

Command: CHKDSK

Cause: A drive has been specified that is redirected over the network. CHKDSK can only operate on local disk drives.

Action: Do not specify a disk drive that is not local.

Cannot do binary reads from a device

Command: COPY

Cause: An attempt was made to copy a file from a device and the /B option was 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.

Action: Re-enter the command line and omit the /B option or specify the /A option after the device name in the command line.

Cannot edit BAK file—rename file

Command: EDLIN

Cause: The filename specified to edit has an extension of .BAK.

Action: Rename the file giving it a different extension, or copy the file to a new file with an extension other than BAK.

Cannot FORMAT nonremovable drive d:

Command: BACKUP

Cause: You used BACKUP with /F and the destination disk is a hard disk.

Action: Try again without the switch, or use flexible disks for the backup.

Cannot format an ASSIGNed or SUBSTed drive

Command: FORMAT

Cause: A drive with an active assignment was specified as the target drive.

Action: Use an ASSIGN command to clear the assignment and re-enter the FORMAT command.

Cannot FORMAT a Network drive

Command: FORMAT

Cause: A drive was specified that is redirected over the network.

Action: Do not specify a non-local drive.

Cannot recover . entry, processing continued

Command: CHKDSK

Cause: The active directory (indicated by the single period) is defective.

Action: No action required.

Cannot RECOVER a Network drive

Command: RECOVER

Cause: A drive has been specified that is redirected over the network.

Action: Do not specify a drive that is not local.

Cannot recover directory entry, processing continued

Command: CHKDSK

Cause: CHKDSK cannot recover *directory* (the current or parent directory) in *directoryname* (the directory having damage).

Action: Copy files from the directory as possible, erase the files in the subdirectory, and remove the subdirectory. Then, restore the files from your backup (See RESTORE and BACKUP before you do anything.).

Cannot recover .. entry

Command: CHKDSK

Cause: The parent directory (indicated by the dual periods) is defective.

Action: No action required.

Cannot use FASTOPEN for drive d:

Command: FASTOPEN

Cause: You used FASTOPEN on a drive affected by an ASSIGN, JOIN, or SUBST.

Action: Undo the effects of the appropriate command, and try again.

CHDIR .. failed trying alternate method

Command: CHKDSK

Cause: CHKDSK became confused and could not get to a parent directory.

Action: Restart DOS and rerun CHKDSK. If the command still fails, the disk has a serious flaw. Retire a flexible disk. Try to reformat a hard disk.

Codepage not prepared

Command: MODE

Cause: For MODE /STATUS, a space is available, but no code page exists for the space. For a MODE SELECT, the selected code page is not prepared. On using MODE for the console device named CON:, the selected code page might not have the font for the current video mode.

Action: To solve this problem, use MODE with PREPARE to reload the appropriate code page and re-execute MODE with SELECT. On getting the error again, edit the DEVICE=DISPLAY.SYS line, increasing the value of *subfonts*. Then, restart DOS.

Codepage operation not supported on this device

Command: MODE

Cause: On executing MODE with either PREPARE or SELECT, one of the following things happened. You:

1. entered an incorrect device;
2. misspelled *devicename;*
3. entered a nonexisting device name;
4. entered a device name not having an appropriate device driver (DISPLAY.SYS for the console, PRINTER.SYS for the printer);
5. entered the device driver directive incorrectly; or
6. named a device that cannot switch code pages.

Action: To solve the problem:

1. Check the device name specified with the MODE command.
2. If the device name is correct, check the phrasing for the device driver in CONFIG.SYS. This is the:
    a. DISPLAY.SYS line when the problem is in the CON device.
    b. PRINTER.SYS line when the problem is in the LPTx or PRN devices.
3. Check for specifying the appropriate number for LPT for your printer.

If you change CONFIG.SYS, restart DOS and try the MODE command again.

Code pages cannot be prepared

Command: MODE

Action: You used MODE with PREPARE and either:

1. specified more code pages than the device driver allows; or
2. duplicated a code page for the Quietwriter III printer.

Mode does not prepare code pages. It deletes previously prepared code pages having the same position as the code pages in your command.

Action: The Quietwriter III cannot duplicate a hardware code page. For other devices, use MODE with /STATUS to check the allowed number of prepared code pages. To have more code pages than those listed in the prepared code page lines, edit *added_codepages* in the DISPLAY.SYS or PRINTER.SYS lines in CONFIG.SYS. Restart DOS and try the command again.

Code page requested (codepage) is not valid for given keyboard code

Command: KEYB

Cause: You specified a keyboard code, but not a code page.

Action: Try again, specifying a compatible key code and code page.

Code page specified is inconsistent with the selected code page

Command: KEYB

Cause: You specified a key code and code page, but a different code page is active for CON (the console).

Action: Run KEYB to see the current situation. Then, use the MODE CON CODEPAGE SELECT command to activate the appropriate code page.

Code page specified has not been prepared

Command: KEYB

Cause: Your CONFIG.SYS file has a DEVICE=DISPLAY.SYS directive, but the keyboard code used with KEYB needs a code page that is not yet prepared.

Action: Run KEYB without parameters to see the current situation. Then, use a MODE CON CODEPAGE PREPARE command to prepare the code page for the keyboard code you want to use.

Compare error on side x, track y

Command: DISKCOMP

Cause: One or more locations on the indicated side and track of the two disks contain differing information.

Action: You may not need to do anything. The differences may be due to previously copying files. Commands and programs may work fine and merely be stored on differing locations. Running CHKDSK can indicate if one or both disks contain bad locations.

COMy bbb,p,d,s,t initialized

Command: MODE

Cause: Your command initialized the asynchronous communications adapter: *y* is the adapter number; *bbbb* is the baud rate; *p* is the parity; *d* is the number of data bits; *s* is the number of stop bits; and *t* is the retry on a time-out.

Action: None required

Contains invalid cluster, file truncated

Command: CHKDSK

Cause: The FAT has a bad pointer to *filename*.

Action: If you did not use /F, rerun CHKDSK with this option.

Contains xxx noncontiguous blocks

Command: CHKDSK

Cause: A disk has *filename* stored in *xxx* noncontiguous pieces. This is not serious, but it slows performance.

Action: For a flexible disk, copy the files to another flexible disk. If many files have this condition on a hard disk, backup the disk, reformat the disk, and then restore the files.

Content of destination lost before copy

Command: COPY

Cause: The same file has been specified as both a source and the destination drive. The following command shows an example:

```
COPY FILE1+FILE2 FILE1
```

FILE1 is overwritten before it can be copied.

Action: Copy to a temporary file, delete the duplicate file, and rename the temporary file to the desired filename.

Convert directory to file (Y/N)?

Command: CHKDSK

Cause: A directory contains so much bad information, it cannot be used as a directory.

Action: Answer with N on first seeing the question so CHKDSK will take no action. Later, you can copy files from the directory to another disk and see if you can use the files. Then, rerun CHKDSK to convert the directory into a file and try to recover the rest of the files.

Convert lost chains to files (Y/N)?

Command: CHKDSK

Cause: The command detected lost clusters or chains (multiple clusters).

Action: If you answer Y, each chain will be converted to a file with *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 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.

Copy Another (Y/N)?

Command: DISKCOPY

Cause: The command completed copying a disk.

Action: Answer Y to copy another disk, N to abort the command.

Copy Process Ended

Command: DISKCOPY

Cause: You tried to copy disks of differing densities.

Action: Copy disks of similar densities.

Corrections will not be written to disk

Command: CHKDSK

Cause: An error(s) has been detected on the disk, but the /F option was been specified.

Action: Run CHKDSK again with the /F option.

Current keyboard does not support this Codepage

Command: MODE

Cause: You executed KEYB previously to executing MODE with SELECT. A conflict now exists between the current code page for the keyboard and the code page specified by MODE for the console. The code page for the console goes to the specified code page, but the keyboard code page is unchanged.

Action: To use the new code page on the console, execute KEYB with an appropriate keyboard code because executing KEYB automatically switches the keyboard's code page.

Device error during Xxxxxx

Command: MODE

Causes: The error occurs during a MODE with CODEPAGE. *Xxxxx* can be: *Prepare, Refresh, Select, Status,* or *write of font file to device.* The error could be due to the following things:

- The device does not support code page switching.
- CONFIG.SYS has no line for including a device driver.
- The value of *added_codepages* does not allow more code pages.
- The code page information file is corrupted or nonexistent.
- MODE with PREPARE had no name for the file for the code page.
- The code page number was incorrect.
- The device detected a transmission error.
- The device is turned off or not selected.
- You ran MODE REFRESH ahead of MODE SELECT or CHCP.

Action: Try the following before running MODE CODEPAGE:

1. Use MODE SELECT instead of MODE REFRESH. Specify an appropriate device name, code page number, and full file name for the page file. Make sure the device is ready.
2. Examine the CONFIG.SYS file for the following:
    a. DISPLAY.SYS for the console; PRINTER.SYS for the printer.
    b. Specify appropriate parameters throughout.
    c. Check additional code pages, editing the number as required.
    d. Restart DOS and try MODE CODEPAGE again.

You have a hardware problem if you still get the error.

Device or codepage missing from font file

Command: MODE

Cause: You used MODE PREPARE and specified a possible code page whose information page does not have the font.

Action: All existing code pages for the device disappear. You must then prepare them again. Then, restart DOS and try the command again.

Directory entries adjusted

Command: VDISK (a directive in CONFIG.SYS)

Cause: The number of directory entries specified in the DEVICE=VDISK.SYS command in the CONFIG.SYS file may need adjustment.

Action: Change the number of directory entries as necessary.

Directory is joined

Command: CHKDSK

Cause: A directory on the disk being checked has been joined.

Action: Unjoin the directory and run CHKDSK again.

Directory not empty

Command: JOIN

Cause: An attempt was made to join to a non-empty directory.

Action: Join to a new directory which is empty, or empty the directory and rerun JOIN.

Directory is totally empty, no . or ..

Command: CHKDSK

Cause: A subdirectory does not contain the parent or current directory entries.

Action: Delete the subdirectory with RMDIR and recreate it using MKDIR.

Disk error reading FAT [x:]

Command: CHKDSK

Cause: One of the two File Allocation Tables for drive x has a defective sector in it. DOS automatically uses the good FAT.

Action: Copy the files to another disk and reformat the disk. If the sectors for the FAT are still defective, discard the disk.

Disk error writing FAT [x:]

Command: CHKDSK

Cause: One of the two File Allocation Tables for drive x has a defective sector in it. DOS automatically uses the good FAT.

Action: Copy the files to another disk and reformat the disk. If the sectors for the FAT are still defective, discard the disk.

Disk full. Edits lost

Command: EDLIN

Cause: EDLIN could not save your file due to a lack of disk space.

Action: Delete enough files to make room for your document and re-enter it with EDLIN.

Disk unsuitable for system drive

Command: FORMAT

Cause: Your command detected a bad track on the disk where the system files should reside.

Action: This disk should only be used for data. Use another disk if you want to make a system disk.

Diskette/Drive not compatible

Command: DISKCOMP

Cause: The disks being compared have differing densities.

Action: Compare disks of similar densities.

Divide overflow

Command: DOS

Cause: A program attempted to divide by 0, or a logic error caused an internal malfunction.

Action: Correct error in program. If the program is purchased, contact your dealer.

Do you see the leftmost 0? (Y/N)
Do you see the rightmost 9? (Y/N)

Command: MODE

Cause: You get one of the two messages. Both indicate you are adjusting the display for the color/graphics adapter.

Action: Answer with *Y* once a display is centered according to your needs. Before typing *Y*, type *N* if necessary to shift the display left or right.

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.

Action: Be sure the file being renamed exists, and that the new file name does not exist.

End of input file

Command: EDLIN

Cause: 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.

Action: No action required.

EOF mark not found

Command: COMP

Cause: One or both of the files being compared do not contain End- of-file markers.

Action: No action required.

Entry error

Command: EDLIN

Cause: A command has a syntax error.

Action: Retype the command with the correct syntax.

Entry has a bad [attribute or link or size]

Command: CHKDSK

Cause: An error condition was detected in one of the subdirectory entries. The message is preceded by one or two periods to indicate which entry.

Action: CHKDSK attempts to correct the error when the /F option is specified.

Error during read of font file

Command: MODE

Cause: DOS detected a disk error while reading the information file for a code page during a MODE PREPARE. You probably have a bad sector. The MODE PREPARE aborts and does not change the prepared fonts.

Action: Run RECOVER on the information file. If RECOVER reports bad sectors, reformat or discard the disk. Then, copy *.CPI* from your DOS startup disk to the hard disk and try the command again.

Error in EXE file

Command: Several commands can display this error.

Cause: An EXE program attempted to load and execute, and it has an invalid internal format.

Action: Relink the program or make a new copy to execute. If you are using a purchased program and error persists, contact your dealer.

Error opening logfile

Command: BACKUP

Cause: You: omitted the logfile name, used an invalid file name, gave an incorrect disk drive, or gave an invalid path.

Action: Check your use of the /L switch, and try again.

Error writing to device

Command: Several

Cause: Too much data was sent to a device. DOS was unable to write the data to the specified device.

Action: Send less data to the device.

Errors found, F parameter not specified Corrections will not be written to disk

Command: CHKDSK

Cause: CHKDSK detected an error and the /F option was not specified. Corrective action taken by CHKDSK is not written to the disk.

Action: Rerun CHKDSK with the /F option.

Errors on list device indicate that it may be off-line Please check it

Command: PRINT

Cause: A printer time-out error was detected, or there was a long printer operation such as a form feed. This message is only displayed when the PRINT command in executed.

Action: Turn the printer on, or return it to on-line status if appropriate. If the error occurs during a long printer operation, ignore the message.

EXEC failure

Command: DOS

Cause: An error is found while reading a command, or the FILES directive in the CONFIG.SYS file is set too low.

Action: Increase the value for FILES and restart DOS, or check the integrity of the disk.

Failure to access Codepage Font File

Command: MODE

Cause: MODE PREPARE does not find the specified information file for the code page.

Action: Check the spelling, drive, and pathname. Make necessary corrections and try the command again.

Failure to access device: devicename

Command: MODE

Cause: MODE CODEPAGE could not open a device named *devicename.*

Action: Check the device name. Be sure CONFIG.SYS has the device driver. Ensure that the device is connected and turned on. Then, try the command again.

File allocation table bad

Command: DOS

Cause: A disk may be defective.

Action: Run CHKDSK to check the disk.

File allocation table bad drive (x:)

Command: CHKDSK, PRINT

Cause: A disk may be defective.

Action: Run CHKDSK to check the disk.

File cannot be copied onto itself

Command: DOS

Cause: The specified source is the same as the destination. The following example shows this:

```
COPY A A
```

Action: Check what you want to do, and run the command again.

File cannot be converted

Command: EXE2BIN

Cause: The input file is not in the correct format.

Action: Execute the command again with an option that accounts for the desired format. If you get the error again, the file cannot be converted.

File Creation Error

Command: DOS

Cause: You tried to add a filename or replace a file that already exists in the directory, or when you tried to edit a protected file (see PROTECT). If the file already exists, it is a read-only file and cannot be replaced.

Action: Run CHKDSK on the disk to determine the cause.

File is READ-ONLY

Command: EDLIN

Cause: You tried to change a file that is designated read-only.

Action: Change the permission for the file, and then rerun EDLIN to make changes.

(filename) file not found

Command: PRINT

Cause: Disks were switched while a file was queued up, but it had not started to print.

Action: Reissue the PRINT command for that filename.

(filename) is cross linked on cluster

Command: CHKDSK

Cause: Two or more files are cross linked.

Action: Make a copy of the file you want to keep, and then delete files that are cross linked.

Filename must be specified

Command: EDLIN

Cause: You did not specify a filename at the time you started EDLIN.

Action: Specify a file to edit in the command line to start EDLIN.

File not found

Command: EDLIN, FC, FIND, MORE, DOS, RECOVER, EXE2BIN, REN

Cause: DOS cannot find the specified file.

Action: Check your command and try again.

Files are different sizes

Command: COMP

Cause: Two files are different sizes. The compare operation terminates.

Action: No action required.

Find: File not found <file name>

Command: FIND

Cause: You specified a file that does not exist.

Action: Make sure you type the filename correctly, and try again.

Find: Invalid number of parameters

Command: FIND

Cause: You specified too many, or not enough, options in the command line.

Action: Re-enter the command correctly.

Find: Invalid Parameter

Command: FIND

Cause: You specified an incorrect switch.

Action: Re-enter the command correctly.

Find: Read error in filename

Command: FIND

Cause: The program could not read the specified file.

Action: Run CHKDSK to ensure the integrity of the disk and try again.

Find: Syntax error

Command: FIND

Cause: You typed a command incorrectly.

Action: Check the syntax, and try again.

First cluster number is invalid, entry truncated

Command: CHKDSK

Cause: 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.

Action: No action required.

Fix-ups needed - base segment (hex:)

Command: EXE2BIN

Cause: The source (EXE) file contained information indicating that a load segment is required for the file.

Action: Specify the absolute segment address at which the finished module is to be located.

Font file contents invalid

Command: MODE

Cause: MODE PREPARE found the specified code page file but the file has inappropriate contents. The information file for the code page has been destroyed, or you specified a name of an existing file and the file is not an information file for a code page.

Action: You lose prepared code pages for the device, so copy the information file for the code page from your DOS startup disk. Make sure it has appropriate contents, and try the command again.

For cannot be nested

Command: BATCH COMMAND PROCESSOR

Cause: DOS displays the message when you nest FOR commands in a batch file.

Action: Check what you want to do, and develop appropriate logic and commands for the file.

Format failure

Command: FORMAT

Cause: DOS could not format the disk. The message is usually displayed with an explanation as to why DOS could not format the disk.

Action: Restart FORMAT with a new disk.

Formatting While Copying

Command: DISKCOPY

Cause: The destination disk is not recognized as a formatted disk. DISKCOPY will format while it copies from the source disk.

Action: No action required.

Illegal Device Name

Command: MODE

Cause: You failed to specify a number with LPTx: or COMy:. You specified a number outside the allowed range (1-3 for LPTx:; 1, 2 for COMy:; 1-3 for PC computers; 4 for computers running PS/2). You omitted a space between LPTx: or COMy: and the next parameter. You inserted more than one space between LPTx: and COMy: and the next parameter. Or, your system does not have the specified adapter.

Action: Correct the problem, and try again.

Incorrect DOS version

Command: Many commands can display this message.

Cause: Many versions of DOS 2.00 utilities do not run on other versions of DOS.

Action: Use the appropriate utilities with the appropriate version of DOS.

Incorrect number of parameters

Command: JOIN, SUBST

Cause: You specified too many or too few options in the command line.

Action: Re-enter the command line correctly.

Incorrect parameter

Command: SHARE

Cause: You specified an incorrect option.

Action: Re-enter the command line correctly.

Infinite retry on parallel printer timeout
No retry on parallel printer timeout

Command: MODE

Cause: You used MODE with LPTx: or COMy:. Specifying the *P* option gives the first message; not specifying *P* gives the second message.

Action: None required.

Insert diskette for drive (x)
and strike any key when ready

Command: DOS

Cause: DOS is copying or formatting.

Action: Insert a disk in the appropriate drive and press an alphanumeric key to begin processing.

Insert disk with batch file and press any key when ready

Command: DOS

Cause: The disk containing a specified batch file is not in the originally specified drive.

Action: Reinsert the disk containing the batch file in the appropriate drive.

Insert DOS diskette into drive (x) and strike any key when ready

Command: FORMAT

Cause: You specified FORMAT /S, but the disk in the default drive does not contain DOS system files.

Action: Insert a disk with the files IO.SYS and DOS.SYS in the specified drive.

Insufficient disk space

Command: DOS, SORT

Cause: The disk is full. It does not contain enough room to perform the specified operation.

Action: Delete extraneous files from the disk, and try again.

Insufficient memory for system transfer

Command: FORMAT

Cause: The memory configuration is insufficient to transfer the DOS system files IO.SYS and MSDOS.SYS (the /S switch)

Action: Re-boot the system with fewer device drivers, virtual disks, or more memory, and then try again.

Insufficient room in root directory Erase files in root and repeat CHKDSK

Command: CHKDSK

Cause: CHKDSK always recovers lost files into the root directory. In this case, your root directory is full.

Action: Delete unused files in the root directory, and try again.

Intermediate file error during pipe

Command: DOS

Cause: The pipe operation makes use of temporary files on the disk which are automatically deleted after a piping process is completed. An error has occurred in one of these files.

Action: Check the integrity of your disk with CHKDSK. Try again with a new disk.

Invalid baud rate specified

Command: MODE

Cause: You specified a baud rate other than 110, 150, 300, 600, 1200, 2400, 4800, or 9600.

Action: Re-enter command line with valid baud rate.

Invalid characters in volume label

Command: FORMAT

Cause: The volume label contains invalid, or too many, characters.

Action: Use a LABEL command to ensure that the volume label contains no more than 11 alphanumeric characters.

Invalid code page specified

Command: KEYB

Cause: You specified a nonexistent code page.

Action: Determine the appropriate value, and try again.

Invalid COMMAND.COM.

Command: DOS

Cause: The program you have just run has used up almost all of memory. DOS must now reload the COMMAND.COM file from disk. However, DOS cannot find COMMAND.COM on the disk, or the copy it found is invalid.

Action: Insert a disk into the default drive that contains a copy of the COMMAND.COM version you used to start DOS.

Invalid country code

Command: DOS

Cause: You specified a country number in you CONFIG.SYS file that is not in the table of countries configured in this implementation of DOS.

Action: See the COUNTRY command for a list of country codes.

Invalid keyboard code specified

Command: KEYB

Cause: You specified a nonexistent key code.

Action: Determine the appropriate value, and try again.

Invalid current directory

Command: CHKDSK

Cause: A directory contains invalid information.

Action: When CHKDSK ends, change to the faulty directory and run the following command to get more information about what is wrong:

```
CHKDSK *.* /V
```

Then, run CHKDSK with /F to fix the problem.

Invalid date

Command: DOS, DATE

Cause: You specified an invalid date in response to the date prompt when starting DOS.

Action: Retype date correctly.

Invalid device

Command: DOS

Cause: The specified device was not CON, NUL, AUX, or PRN.

Action: Try again with a valid device name.

Invalid directory

Command: DOS

Cause: The specified directory either does not exist or is invalid, or you tried to access a protected directory (see PROTECT).

Action: Enter the directory name correctly.

Invalid drive in search path

Command: DOS

Cause: An invalid drive was detected in the PATH string.

Action: Issue a new PATH command with no invalid drives in it.

Invalid drive or filename

Command: EDLIN, RECOVER

Cause: You need to specify a valid file name.

Action: Specify a drive that is present.

Invalid drive specification

Command: Many commands display this message

Cause: This error message is displayed by CHKDSK, FORMAT or SYS when a valid drive needs to be specified.

Action: Specify a valid drive ID.

Invalid drive specification
Specified drive does not exist, or is non-removable.

Command: DISKCOPY, DISKCOMP

Cause: You specified a drive designator for a drive that does not exist or a drive that is non-removable.

Action: Reissue the command with an existing, removable drive.

Invalid number of parameters

Command: FIND, DOS, RECOVER

Cause: This error message is displayed by FC, FIND or RECOVER when there is a wrong number of options in the command line.

Action: Reissue the command with the correct number of options.

Invalid parameter

Command: Many commands display this message

Cause: This error message is displayed by CHKDSK, EDLIN, FC, FORMAT or PRINT when one of the specified switches is wrong.

Action: Check the syntax, and reissue the command.

Invalid parameter
Do not specify filename(s).
Command Format: DISKCOMP d: d: [/1][/8]

Command: DISKCOMP

Cause: One or more filenames were encountered. DISKCOMP only accepts drive designators.

Action: Reissue the command using an appropriate syntax.

Invalid parameter
Do not specify filename(s).
Command Format: DISKCOPY d: d: [/1]

Command: DISKCOPY

Cause: One or more filenames were encountered. DISKCOPY only accepts drive designators.

Action: Re-enter the command using an appropriate syntax.

Invalid path or file name

Command: DOS

Cause: DOS displays the message when a valid pathname or filename to the COPY command needs to be specified, or when you try to access a protected file (see PROTECT).

Action: Run the command again with an appropriate name.

Invalid path, not directory, or directory not empty

Command: RMDIR

Cause: You are unable to remove the directory requested for one of the specified reasons, or you tried to remove a protected directory (see PROTECT).

Action: Be sure the directory name exists and is empty, then reissue the command.

Invalid sub-directory entry

Command: CHKDSK

Cause: The specified subdirectory does not exist, or it is invalid.

Action: Check to see that you typed the subdirectory name correctly.

Invalid switch character

Command: VDISK directive

Cause: The character following the "/" in the command line was not "E". VDISK will attempt to install the virtual disk in low memory.

Action: Re-enter command line in CONFIG.SYS file.

Invalid time

Command:	DOS, TIME
Cause:	You specified an invalid time in response the TIME prompt.
Action:	Enter a valid time in the correct (24-hour) format.

Label not found

Command:	DOS batch file processor
Cause:	DOS displays the message when there is a GOTO command to a nonexistent label in a batch file.
Action:	Edit the batch file so the label exists, or remove the reference to the label.

Last file not backed up

Command:	BACKUP
Cause:	The hard disk is full.
Action:	Delete unnecessary files, and try again, or back up on flexible disks.

Line too long

Command:	EDLIN
Cause:	The replacement string in a Replace command caused the line to expand beyond 253 characters.
Action:	Divide the long line into two lines and retry the Replace command.

List output is not assigned to a device

Command: PRINT

Cause: PRINT displayed the message when you first the command and you were asked to specify the device to be used as the as the print spooler. This message also appears if PRINT is set up for a device that does not exist.

Action: Specify a device name that exists in your system.

Memory allocation error

Command: DOS

Cause: A program has written into "free" memory, preventing DOS from allocating that memory. The error is usually fatal.

Action: Restart DOS, and if the error persists, make a new copy of the DOS disk from your backup copy of the system disk. If this error occurs consistently with purchased software, see your dealer.

—More—

Command: MORE

Cause: The program which has piped its output to MORE has produced more than one screen of data.

Action: Press the spacebar to view more of the file or directory.

MORE: Incorrect DOS version

Command: MORE

Cause: MORE does not run on versions of DOS prior to 2.0.

Action: Use the correct version of MORE and DOS.

Must specify destination line number

Command: EDLIN

Cause: You must specify a destination line number when copying and inserting lines with EDLIN.

Action: Reissue the command, including the destination line number.

Must specify ON or OFF

Command: DOS

Cause: The command requires either an ON or an OFF argument.

Action: Reissue the command, specifying either ON or OFF.

Name of list device (PRN):

Command: PRINT

Cause: This prompt appears the first time that PRINT is run, requesting the PRINT output device.

Action: Specify the device you want to print on.

New file

Command: EDLIN

Cause: EDLIN did not find a file with the name you specified.

Action: 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.

No codepage has been SELECTED

Command: MODE

Cause: You executed MODE /STATUS and have not yet executed MODE SELECT on a device.

Action: Run MODE SELECT, and try again.

No free file handle
Cannot start COMMAND, exiting

Command: DOS

Cause: COMMAND.COM could not open a file because there are not enough file handles available in the system.

Action: Restart DOS. Add or increase the number in the FILES statement in CONFIG.SYS. Then restart DOS again.

No path

Command: PATH

Cause: There is no PATH variable set in the environment.

Action: No action required.

No room in directory for file

Command: EDLIN

Cause: You tried to save a file to the root directory but it is full. Subdirectories are not limited in size as is the root directory.

Action: Delete extraneous files from the root, or specify a file name in a subdirectory. Re-enter your text into EDLIN.

Not enough memory

Command: JOIN, SHARE, SUBST

Cause: There is not enough memory for DOS to run the command.

Action: Reduce the number of installed drivers and virtual disks, or acquire more memory for your system.

Not enough room for DOS on this disk

Command: SYS

Cause: There is not enough room for the system files on the destination disk.

Action: Delete some files to make room for the system files or use another disk. You may need to reformat the disk to put the system on it.

Not enough room to merge the entire file

Command: EDLIN

Cause: There is not enough room in memory to hold the file during a Transfer command.

Action: Free some memory by writing some of the text to a disk before reissuing the Transfer command.

Not found

Command: EDLIN

Cause: You specified a Search or Replace command that was unable to find another occurrence of the specified Search or Replace string.

Action: No action required.

O.K.?

Command: EDLIN

Cause: This prompt occurs during Search and Replace command processing.

Action: If you press any key except Y or [Return], the search or replace process continues.

One or more CON code pages invalid for given keyboard code

Command: KEYB

Cause: You prepared code pages for the console, but the specified keyboard code is not compatible with them.

Action: Examine the code pages to note what is wrong with them. Then, make changes that let the code page work with the keyboard code.

Out of environment space

Command: SET, PATH, PROMPT

Cause: There is not enough room in the program environment to accept more data.

Action: Remove unnecessary strings from the environment. Reissue the command.

Path not found

Command: CHKDSK

Cause: You specified an invalid path name.

Action: Check your command, and try again.

Port not installed

Command: MODE

Cause: The specified port (LPTn: or COMn:) is not present in the system.

Action: Specify a port that is in the system.

Press any key to begin recovery of the (xxx) file(s) on drive (x:)

Command: RECOVER

Cause: This prompt appears before you recover a disk or file.

Action: Press a key to begin the recover process. Use Ctrl-Break to end the process.

PRINT queue is full

Command: PRINT

Cause: There is no room in the list of files waiting to be printed.

Action: Wait until some of the waiting files have been printed, or remove some of them from the queue. Reissue the PRINT command.

Probable non-DOS disk
Continue (Y/N)?

Command: CHKDSK

Cause: The disk being used is not recognized by this version of DOS. The disk was either created by another system with a format that is not supported on this version of DOS or is not an DOS disk.

Action: If the /F parameter was used, reissue the command without /F. This displays possible corrections. The disk may then be recovered with /F, or reformatted.

Processing cannot continue

Command: CHKDSK

Cause: There is not enough memory in your machine to process CHKDSK for this disk.

Action: Remove some drivers or virtual disks, or obtain more memory for your system.

Program is too big to fit in memory

Command: DOS

Cause: You must acquire more memory to run your application. It is possible that some programs you have run are still using some memory.

Action: You can try to restart DOS, but if you still get the message, you must acquire more memory.

Resident part of PRINT installed

Command: PRINT

Cause: Available memory has been reduced by several thousand bytes to process the print command concurrent with other processes.

Action: No action required.

Re-insert diskette for drive x

Command: FORMAT

Cause: You should reinsert the disk being formatted in the indicated drive.

Action: Insert the disk to be formatted in the specified drive.

Resync failed. Files are too different

Command: FC

Cause: After FC has detected many differences, it gives up and displays this message.

Action: Edit the files to remove the reported differences and try again.

Sector size adjusted

Command: VDISK

Cause: It is necessary to adjust the sector size specified in the DEVICE=VDISK.SYS command in the CONFIG.SYS file.

Action: Change sector size.

Sector size too large in file (filename)

Command: -DOS

Cause: 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.

Action: No action required.

SHARE already installed

Command: SHARE

Cause: You tried to install SHARE more than once.

Action: No action required.

SORT: Incorrect DOS version

Command: SORT

Cause: You tried to run SORT on an incorrect version of DOS.

Action: Use the correct version of SORT and DOS.

SORT: Insufficient disk space

Command: SORT

Cause: The disk is full.

Action: Remove some extraneous files from the disk, and try again.

SORT: Insufficient memory

Command: SORT

Cause: There is not enough memory to run the SORT program.

Action: Reduce the amount of data to be sorted.

SOURCE diskette bad or incompatible

Command: DISKCOMP, DISKCOPY

Cause: One of the disks is unformatted or incompatible with the drive.

Action: Use the a properly formatted disk.

Specified COMMAND search directory bad

Command: DOS

Cause: The SHELL command in the CONFIG.SYS file is incorrect. The place that you have told DOS to find COMMAND.COM does not exist, or COMMAND.COM is not in that place.

Action: Start the SHELL program from the root directory.

Strike a key when ready . . .

Command: 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.

Action: Perform the specified task, then press a key.

Syntax error

Command: FIND

Cause: You typed a command line that FIND cannot interpret.

Action: Check the syntax, and try again.

Target cannot be used for backup

Command: BACKUP

Cause: The destination disk has a problem.

Action: For a flexible disk, try a different disk. For a hard disk, run CHKDSK, and try again if the disk is alright.

Target diskette is write protected Correct, then strike any key

Command: DISKCOPY

Cause: The target disk is write-protected.

Action: Write-enable the disk and reinsert it.

Target Diskette may be unusable

Command: DISKCOPY

Cause: Read, write, or verify errors occurred during the copying process. The target disk may be incomplete.

Action: Compare the two disks with DISKCOMP. Try again. You might need a new destination disk.

Terminate batch job (Y/N)?

Command: DOS

Cause: You pressed Ctrl-K while you were in batch mode. DOS asks if you want to end batch processing.

Action: Press Y to end processing. Press N to continue the batch job.

Too many files open

Command: EDLIN

Cause: DOS could not open the BAK file, or a file to edit, due to lack of system file handles.

Action: Increase the value of the FILES command in the CONFIG.SYS file.

Track O bad - disk unusable

Command: FORMAT

Cause: The FORMAT command cannot accommodate defective sectors on the disk.

Action: Format another disk.

Unable to create directory

Command: DOS, MKDIR

Cause: DOS could not create a specified directory, or could not create a subdirectory in a protected directory (see PROTECT).

Action: Check for a name conflict (you may have a file by the same name). The disk may be full.

Unable to create KEYB table in resident memory

Command: KEYB

Cause: KEYB installs into a predetermined amount of memory. You added code pages to the console (using other commands), and you have now attempted to use KEYB again.

Action: To use additional code pages, restart DOS and run the MODE commands for code pages before you run KEYB.

Unrecognized command in CONFIG.SYS

Command: CONFIG.SYS

Cause: You have an invalid directive in the CONFIG.SYS file.

Action: The chapter named "System Configuration" in this manual lists valid directives.

Unrecoverable error in Directory Convert directory to file (Y/N)?

Command: CHKDSK

Cause: There has been an unrecoverable error in directory.

Action: If you type Y, CHKDSK converts the bad directory into a file. You can then fix the directory yourself, or delete it.

Unrecoverable Read Error on Drive x: Side y, Track z

Command: DISKCOMP, DISKCOPY

Cause: You get the message after several unsuccessful attempts are made to read the data from the specified track and sector on the disk in the indicated drive.

Action: You can RECOVER as many files as possible, copy the files to another disk, and try to reformat the disk.

Unrecoverable Write Error on Drive x: Side y, Track z

Command: DISKCOPY

Cause: You get this message after several unsuccessful attempts are made to write data to the specified track and sector on the disk in the indicated drive.

Action: You can RECOVER as many files as possible, copy the files to another disk, and try to reformat the disk.

VDISK not installed - buffer too small

Command: VDISK

Cause: The virtual disk drive cannot be installed due to an incorrect buffer size.

Action: Change buffer size.

VDISK not installed - insufficient memory

Command: VDISK

Cause: The virtual disk drive cannot be installed due to insufficient memory. The system has insufficient memory when less than 64 Kbytes of system memory would be left after the virtual disk is installed.

Action: Change buffer size.

VDISK not installed - no extended memory

Command: VDISK

Cause: You specified the /E switch, but the system does not have extended memory, or the amount of available extended memory is insufficient to contain the virtual disk even after adjusting the parameters.

Action: Ensure that you have enough extended memory to run VDISK, or reduce the size of VDISK.

Volume label (ll characters, ENTER for none)?

Command: FORMAT

Cause: You specified the /V switch in the FORMAT command.

Action: Specify a volume label or press Return to indicate that you do not want a volume label for the disk.

WARNING, ALL DATA ON NON-REMOVABLE DISK DRIVE X WILL BE LOST! Proceed with Format (Y/N)?

Command: FORMAT

Cause: There is data on the hard disk you are trying to format.

Action: If you want to lose the data and format the disk, press Y (for "Yes"). If you do not want the files on your hard disk erased press N (for "No"). Copy the files to a flexible disk and repeat the FORMAT command.

Warning - directory full

Command: RECOVER

Cause: The root directory is too full for RECOVER processing.

Action: Delete some files in the root directory to free space.

Warning! No files were found to back up

Command: BACKUP

Cause: No files on the source disk matched your specification.

Action: Run DIR to list your files. Then, try again with an appropriate specification.

Warning: Read error in EXE file

Command: EXE2BIN

Cause: The amount of data read was less than the size of the header. This is a warning message only.

Action: None required.

B

Configuration Command Reference

Overview

MS-DOS includes special configuration commands that can be 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. With the exception of BREAK, configuration commands can only be used within *config.sys*.

MS-DOS also allows you 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. Four device drivers are provided with your system: VDISK.SYS, ANSI.SYS, PEMM.SYS and NETDRVR.SYS.

The configuration commands and supplied device drivers are detailed in the remaining pages of this chapter. For instructions for editing and using *config.sys*, see your *Using the Series 300 DOS Coprocessor* or *SoftPC User's Guide* manual.

ANSI.SYS

Purpose

ANSI.SYS is a device driver that can be loaded with the DEVICE command. The ANSI.SYS driver supports extended keyboard and screen display features.

Syntax

```
DEVICE = ANSI>SYS
```

Operation

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.*

BREAK

Purpose

The BREAK command enables or disables extended [CTRL]-[Break] and [CTRL]-[C] checking.

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 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; however, the *config.sys* file provided with the DOS Coprocessor software includes the command:

```
BREAK = ON
```

which enables the extended [CTRL]-[Break] and [CTRL]-[C] 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.

Many application programs tell you how to set the BUFFERS command for optimum performance.

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

Selects 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 country code for each country is listed below.

Country	Code	Country	Code
United States	001	Netherlands	031
Belgium	032	France	033
Spain	034	Italy	039
Switzerland	041	United Kingdom	044
Denmark	045	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 loads a file containing an installable device driver. The driver is incorporated into the resident portion of MS-DOS.

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. Four installable device drivers are provided with the DOS Coprocessor. They are VDISK.SYS, which provides the ability to use system RAM as a virtual disc, ANSI.SYS, which provides ANSI display terminal emulation for the standard output device, PEMM.SYS which provides access to expanded memory compatible with the Lotus/Intel/Microsoft Expanded Memory Specification, which provides support for accessing the HP-UX file system via drive D:.

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 command 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 least recently used. If an application program attempts to open more files than specified and file-sharing has been loaded, MS-DOS will close the least 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 by 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 letter for the system. The command line:

```
LASTDRIVE = G
```

in the *config.sys* file will provide 7 disc drive letters, 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 increase the number of valid drive letters.

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.

PEMM.SYS

Purpose

The DEVICE command loads PEMM.SYS, a device driver that provides for accessing the expanded memory compatible with the Lotus/Intel/Microsoft Expanded Memory Specification (version 4.0).

Syntax

`DEVICE=PEMM.SYS` */H:nnn* */V*

Switch provides verbose information

Provides a handle to programs

Operation

The examples shown in the table illustrate how to use the command. The following Notes provide additional information.

Command	Result/Action
`device = pemm.sys /H:100 /V`	Installs the support provided by PEMM.SYS. The */H:100* sets up 100 handles, and the */V* instructs the driver to print information about the configuration at initialization time.

Notes

- This driver runs on DOS Coprocessor only.
- The */H:nnn* option sets up handles for programs. Not using the switch defaults to 64 handles, which is adequate for most situations (you should not change the value unless you know why you need to change it). The range for *nnn* is 32 to 255. Each handle uses 24 bytes of memory. Using a value less than the default reduces the size of the installed driver, and conversely. Each program that uses expanded memory uses one handle. An environment manager that runs multiple programs may open more than one handle.
- The */V* option instructs the driver to print configuration information at initialization time. Using the option lets you see what happens.
- Installing PEMM.SYS lets you allocate expanded memory in the EXPMEM line of DOS.CNF or with the *-expm* option in the DOS startup command line (for example, *DOS -expm*).
- The LIM Expanded Memory Specification lets you allocate more than the standard limit of 640 Kbytes of memory for use with specific applications (for example, LOTUS 123). In this example, allocating more memory lets you work with larger spreadsheets, among other things.
- The driver allocates memory from your computer system's virtual memory. Therefore, the allocated memory cannot exceed the memory available in your system's swap space.

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.

VDISK.SYS

Purpose

VDISK.SYS is a device driver that can be loaded with the DEVICE command. The VDISK block device driver allows a portion of the computer's RAM 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 are much faster than physical disc drives, but their contents will be lost when the system is reset or loses power.

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.

Syntax

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

Operation

VDISK's 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.

If you specify the parameter, and the memory is not present, VDISK.SYS issues an error message and the VDISK is not installed. See "Configuring Memory" in the configuration chapter of *Using the Series 300 DOS Coprocessor*.

<*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.

```
DISK 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>
```

Notes

1. You *must* place DEVICE = VDISK.SYS lines following all other DEVICE lines in the *config.sys* file.
2. This driver is not supported on SoftPC.

C

HP-UX Commands

This appendix contains HP-UX reference pages for DOS-related HP-UX commands.

This page has no reference information.

NAME

dos2ux, ux2dos – convert ASCII file format

SYNOPSIS

dos2ux *file* ...
ux2dos *file* ...

DESCRIPTION

Dos2ux and *ux2dos* read each specified *file* in sequence and write it to standard output, converting to HP-UX format or to DOS format, respectively. Each *file* can be either DOS format or HP-UX format for either command.

A DOS file name is recognized by the presence of an embedded colon (:) delimiter; see *dosif*(4) for DOS file naming conventions.

If no input file is given or if the argument – is encountered, *dos2ux* and *ux2dos* read from standard input. Standard input can be combined with other files.

EXAMPLES

The following prints file **myfile** on the display:

dos2ux myfile

The following converts **file1** and **file2** to DOS format, then concatenates them together, placing them in **file3**.

ux2dos file1 file2 > file3

RETURN VALUE

Both commands return **0** if successful or **2** if the command failed. The only possible failure is the inability to open a specified file, in which case the commands print a warning.

WARNINGS

Command formats resembling:

dos2ux file1 file2 > file1

overwrite the data in **file1** before the concatenation begins, causing a loss of the contents of **file1**. Therefore, be careful when using shell special characters.

SEE ALSO

doschmod(1), doscp(1), dosdf(1), dosls(1), dosmkdir(1), dosrm(1), dosif(4).

NAME

doschmod – change attributes of a DOS file

SYNOPSIS

doschmod [**–u**] *mode device*:*file* ...

DESCRIPTION

Doschmod is the DOS counterpart of *chmod*(1).

There is one option:

–u Disable argument case conversion. In the absence of this option, all DOS file names are converted to upper case.

A DOS file name is recognized by the presence of an embedded colon (:) delimiter; see *dosif*(4) for DOS file naming conventions.

The attributes of each named file are changed according to *mode*, which is an octal number in the range 000 to 0377. *Mode* is constructed from the logical OR of the following modes:

200	Reserved. Do not use.
100	Reserved. Do not use.
040	Archive. Set whenever the file has been written to and closed.
020	Directory. Do not modify.
010	Volume Label. Do not modify.
004	System file. Marks files that are part of the DOS operating system.
002	Hidden file. Marks files that do not appear in a DOS directory listing using the DOS DIR command.
001	Read-Only file. Marks files as read-only.

SPECIAL WARNING

Specifying inappropriate *mode* values can make files and/or directories inaccessible, and in certain cases can damage the file system. To prevent such problems, do not change the mode of directories and volume labels.

Normal users should have no need to use *mode* bits other than 001, 002, and 040.

EXAMPLES

The following marks file **/dev/rfd9122:memo.txt** as a hidden file:

doschmod 002 /dev/rfd9122:memo.txt

The following marks file **driveC:autoexec.bat** read-only:

doschmod 001 driveC:autoexec.bat

SEE ALSO

chmod(1), dos2ux(1), doscp(1), dosdf(1), dosls(1), dosmkdir(1), dosrm(1), chmod(2), dosif(4).

NAME

doscp – copy to or from DOS files

SYNOPSIS

doscp [–**fvu**] *file1 file2*
doscp [–**fvu**] *file1* [*file2* ...] *directory*

DESCRIPTION

Doscp is the DOS counterpart of *cp*(1). *Doscp* copies a DOS file to a DOS or HP-UX file, an HP-UX file to an HP-UX or DOS file, or HP-UX or DOS files to an HP-UX or DOS directory. The last name in the argument list is the destination file or directory.

A DOS file name is recognized by the presence of an embedded colon (**:**) delimiter; see *dosif*(4) for DOS file naming conventions.

The file name – (dash) is interpreted to mean standard input or standard output depending upon its position in the argument list.

Options

There are several options:

–f Unconditionally write over an existing file. In the absence of this option, *doscp* asks permission to overwrite an existing HP-UX file.

–v Verbose mode. *Doscp* prints the source name.

–u Disable argument case conversion. In the absence of this option, all DOS file names are converted to upper case.

Note: Shell metacharacters (*, **?**, and **[...]**) can be used when specifying HP-UX file names, but cannot be used when specifying a DOS file name, because file name expansion is done by the shell and the DOS utilities do not recognize metacharacters.

RETURN VALUE

Doscp returns 0 if all files are copied successfully. Otherwise, it prints a message to standard error and returns with a non-zero value.

EXAMPLES

Copy the files in the HP-UX directory **abc** to the DOS volume stored as HP-UX file **hard_disk**:

doscp abc/* hard_disk:

Copy DOS file **/backup/log** through the HP-UX special file **/dev/rfd9127** to HP-UX file **logcopy** located in the current directory:

doscp /dev/rfd9127:/backup/log logcopy

Copy DOS file **zulu** on the volume stored as HP-UX file **bb** to standard output:

doscp bb:zulu –

SEE ALSO

cp(1), dos2ux(1), doschmod(1), dosdf(1), dosls(1), dosmkdir(1), dosrm(1), dosif(4).

NAME

dosdf – report number of free disk clusters

SYNOPSIS

dosdf *device*[:]

DESCRIPTION

Dosdf is the DOS counterpart of *df*(1). It prints the cluster size in bytes and the number of free clusters on the specified DOS volume.

SEE ALSO

df(1), dos2ux(1), doschmod(1), doscp(1), dosls(1), dosmkdir(1), dosrm(1), dosif(4).

NAME

dosls, dosll – list contents of DOS directories

SYNOPSIS

dosls [–**aAudl**] *device*:[*file*]
dosll [–**aAudl**] *device*:[*file*]

DESCRIPTION

Dosls is the DOS counterpart of *ls*(1).

For each directory named, *dosls* lists the contents of that directory. For each file named, *dosls* repeats its name and any other information requested. If invoked by the name *dosll*, the –**l** option is implied.

Options

There are several options:

–**a** List all directory entries. In the absence of this option, hidden files, system files, and files whose names begin with a dot (.) are not listed.

–**A** Same as –**a**, except the current directory and the parent directory are not listed. For the superuser, this option defaults to being set, and is disabled by –**A**.

–**u** Disable argument case conversion. In the absence of this option, all DOS file names are converted to uppercase.

–**d** If an argument is a directory, list only its name. Often used with –**l** to get the status of a directory.

–**l** List in long format, giving file attribute, size in bytes, and the date and time of last modification for each file, as well as listing the DOS volume label. Long listing is disabled if *dosll* is invoked with the –**l** option.

A DOS file name is recognized by the presence of an embedded colon (:) delimiter; see *dosif*(4) for DOS file naming conventions.

EXAMPLES

These examples assume that a DOS directory structure exists on the device accessed through HP-UX special file **/dev/rdsk/0s1**.

The following example lists all of the files in the root directory of the DOS directory structure:

dosls –a /dev/rdsk/0s1:

The following example produces a long-format listing of all the information about the DOS directory **/dos/math**, but does not list the files in the directory:

dosls –ld /dev/rdsk/0s1:/dos/math

SEE ALSO

dos2ux(1), doschmod(1), doscp(1), dosdf(1), dosmkdir(1), dosrm(1), ls(1), dosif(4).

NAME

dosmkdir – make a DOS directory

SYNOPSIS

dosmkdir [**–u**] *device*:*directory* ...

DESCRIPTION

Dosmkdir is the DOS counterpart of *mkdir*(1). It creates specified directories. The standard entries, . for the directory itself and .. for its parent, are made automatically.

There is one option:

–u Disable argument case conversion. In the absence of this option, all DOS file names are converted to uppercase.

A DOS file name is recognized by the presence of an embedded colon (:) delimiter; see *dosif*(4) for DOS file naming conventions.

DIAGNOSTICS

Dosmkdir returns 0 if all directories were successfully created. Otherwise, it prints a message to standard error and returns non-zero.

EXAMPLES

To create an empty subdirectory named **numbers** under the directory **/math/lib** on the device accessed through HP-UX special file **/dev/rfd9122**, use:

dosmkdir /dev/rfd9122:/math/lib/numbers

SEE ALSO

dos2ux(1), doschmod(1), doscp(1), dosdf(1), dosls(1), dosrm(1), mkdir(1), dosif(4).

NAME

dosrm, dosrmdir – remove DOS files or directories

SYNOPSIS

dosrm [–**friu**] *device*:*file* ...
dosrmdir [–**u**] *device*:*file* ...

DESCRIPTION

Dosrm and *dosrmdir* are DOS counterparts of *rm*(1) and *rmdir*(1), respectively.

Dosrm removes the entries for one or more files from a directory. If a specified file is a directory, an error message is printed unless the optional argument –**r** is specified (see below).

Dosrmdir removes entries for the named directories, provided they are empty.

Options

The options are:

–**f** (force) Unconditionally remove the specified file, even if the file is marked read-only.

–**r** Cause *dosrm* to recursively delete the entire contents of a directory, followed by the directory itself. *Dosrm* can recursively delete up to 17 levels of directories.

–**i** (interactive) Cause *dosrm* to ask whether or not to delete each file. If –**r** is also specified, *dosrm* asks whether to examine each directory encountered.

–**u** Disable argument case conversion. In the absence of this option, all DOS file names are converted to uppercase.

A DOS file name is recognized by the presence of an embedded colon (:) delimiter; see *dosif*(4) for DOS file naming conventions.

EXAMPLES

These examples assume that a DOS directory structure exists on the device accessed through the HP-UX special file **/dev/rfd9122**.

This example recursively combs through the DOS directory **/tmp** and asks if each DOS file should be removed (forced, with no file mode checks):

dosrm –irf /dev/rfd9122:/tmp

The following example removes the DOS directory **doug** from the DOS volume stored as HP-UX file **hard_disk**:

dosrmdir hard_disk:doug

SEE ALSO

dos2ux(1), doschmod(1), doscp(1), dosdf(1), dosls(1), dosmkdir(1), rm(1), rmdir(1), dosif(4).

NAME

DOSIF – DOS Interchange Format description

DESCRIPTION

DOSIF (DOS Interchange Format) is the name given to the media format used by the DOS operating system. This format is based upon that used in IBM PC and PC AT, HP Vectra, and HP 150 systems.

The DOS utilities described in Section 1 (referred to hereafter as *dos**(1)) are provided for reading data from and writing data to DOSIF volumes. Use these utilities to retrieve information from a DOSIF volume.

The *dos**(1) utilities are the only HP-UX commands that can interact directly with the contents of a DOSIF volume. The only other way to interact with the contents of a DOSIF volume is to use an HP-UX DOS emulation or coprocessor facility such as SoftPC or the DOS Coprocessor. *Mount*(1) cannot be used on a DOSIF volume because the operating system does not recognize it.

When constructing file names for *dos**(1) commands, start with the HP-UX path name of the DOSIF volume, then add a colon (:) followed by the file name:

device_file:*file*

or

path_name:*file*

Note: This file naming convention is suitable for use only in arguments to the *dos**(1) utilities. It does not constitute a legal path name for any other use in HP-UX.

Note: Shell metacharacters (*, **?**, and **[...]**) can be used to name HP-UX files, but cannot be used when specifying a DOS file name, because file name expansion is done by the shell and the *dos**(1) utilities do not recognize metacharacters.

By convention, if the HP-UX device name and a trailing colon are specified, but no file or directory name is provided (for example, **/dev/rfd.0:**), the root (/) of the DOS file system is assumed.

EXAMPLES

Specify DOSIF file **/dos/ivy** accessed through HP-UX special file **/dev/rfd9127**:

/dev/rfd9127:/dos/ivy

Specify DOSIF file **/math** accessed through the DOS volume stored as HP-UX file **/users/mydir/driveC**:

/users/mydir/driveC:/math

SEE ALSO

dos2ux(1), doschmod(1), doscp(1), dosdf(1), dosls(1), dosmkdir(1), dosrm(1).

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

HP Part Number
98870-90050
Microfiche No. 98870-99050
Printed in U.S.A. E0989

98870-90651
For Internal Use Only