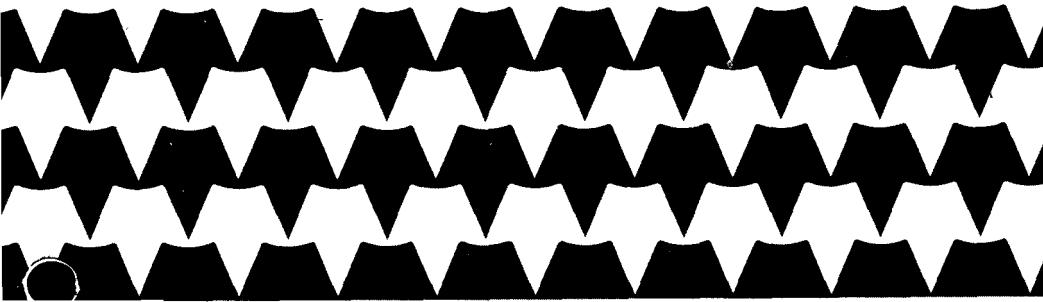


Tandy 3000

MS-DOS

Reference Manual



TANDY®

TERMS AND CONDITIONS OF SALE AND LICENSE OF TANDY COMPUTER EQUIPMENT AND
SOFTWARE PURCHASED FROM RADIO SHACK COMPANY-OWNED COMPUTER CENTERS, RETAIL
STORES AND RADIO SHACK FRANCHISEES OR DEALERS AT THEIR AUTHORIZED LOCATIONS

LIMITED WARRANTY

I. CUSTOMER OBLIGATIONS

- A. CUSTOMER assumes full responsibility that this computer hardware purchased (the "Equipment"), and any copies of software included with the Equipment or licensed separately (the "Software") meets the specifications, capacity, capabilities, versatility, and other requirements of CUSTOMER.
- B. CUSTOMER assumes full responsibility for the condition and effectiveness of the operating environment in which the Equipment and Software are to function, and for its installation.

II. LIMITED WARRANTIES AND CONDITIONS OF SALE

- A. For a period of ninety (90) calendar days from the date of the Radio Shack sales document received upon purchase of the Equipment, RADIO SHACK warrants to the original CUSTOMER that the Equipment and the medium upon which the Software is stored is free from manufacturing defects. **This warranty is only applicable to purchases of Tandy Equipment by the original customer from Radio Shack company-owned computer centers, retail stores, and Radio Shack franchisees and dealers at their authorized locations.** The warranty is void if the Equipment or Software has been subjected to improper or abnormal use. If a manufacturing defect is discovered during the stated warranty period, the defective Equipment must be returned to a Radio Shack Computer Center, a Radio Shack retail store, a participating Radio Shack franchisee or a participating Radio Shack dealer for repair, along with a copy of the sales document or lease agreement. The original CUSTOMER'S sole and exclusive remedy in the event of a defect is limited to the correction of the defect by repair, replacement, or refund of the purchase price, at RADIO SHACK'S election and sole expense. RADIO SHACK has no obligation to replace or repair expendable items.
- B. RADIO SHACK makes no warranty as to the design, capability, capacity, or suitability for use of the Software, except as provided in this paragraph. Software is licensed on an "AS IS" basis, without warranty. The original CUSTOMER'S exclusive remedy, in the event of a Software manufacturing defect, is its repair or replacement within thirty (30) calendar days of the date of the Radio Shack sales document received upon license of the Software. The defective Software shall be returned to a Radio Shack Computer Center, a Radio Shack retail store, a participating Radio Shack franchisee or Radio Shack dealer along with the sales document.
- C. Except as provided herein no employee, agent, franchisee, dealer or other person is authorized to give any warranties of any nature on behalf of RADIO SHACK.
- D. **EXCEPT AS PROVIDED HEREIN, RADIO SHACK MAKES NO EXPRESS WARRANTIES, AND ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE IS LIMITED IN ITS DURATION TO THE DURATION OF THE WRITTEN LIMITED WARRANTIES SET FORTH HEREIN.**
- E. Some states do not allow limitations on how long an implied warranty lasts, so the above limitation(s) may not apply to CUSTOMER.

III. LIMITATION OF LIABILITY

- A. **EXCEPT AS PROVIDED HEREIN, RADIO SHACK SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO CUSTOMER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY "EQUIPMENT" OR "SOFTWARE" SOLD, LEASED, LICENSED OR FURNISHED BY RADIO SHACK, INCLUDING, BUT NOT LIMITED TO, ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR OPERATION OF THE "EQUIPMENT" OR "SOFTWARE," IN NO EVENT SHALL RADIO SHACK BE LIABLE FOR LOSS OF PROFITS, OR ANY INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY BREACH OF THIS WARRANTY OR IN ANY MANNER ARISING OUT OF OR CONNECTED WITH THE SALE, LEASE, LICENSE, USE OR ANTICIPATED USE OF THE "EQUIPMENT" OR "SOFTWARE."** NOTWITHSTANDING THE ABOVE LIMITATIONS AND WARRANTIES, RADIO SHACK'S LIABILITY HEREUNDER FOR DAMAGES INCURRED BY CUSTOMER OR OTHERS SHALL NOT EXCEED THE AMOUNT PAID BY CUSTOMER FOR THE PARTICULAR "EQUIPMENT" OR "SOFTWARE" INVOLVED.
- B. RADIO SHACK shall not be liable for any damages caused by delay in delivering or furnishing Equipment and/or Software.
- C. No action arising out of any claimed breach of this Warranty or transactions under this Warranty may be brought more than two (2) years after the cause of action has accrued or more than four (4) years after the date of the Radio Shack sales document for the Equipment or Software, whichever first occurs.
- D. Some states do not allow the limitation or exclusion of incidental or consequential damages, so the above limitation(s) or exclusion(s) may not apply to CUSTOMER.

IV. SOFTWARE LICENSE

RADIO SHACK grants to CUSTOMER a non-exclusive, paid-up license to use the TANDY Software on **one** computer, subject to the following provisions:

- A. Except as otherwise provided in this Software License, applicable copyright laws shall apply to the Software.
- B. Title to the medium on which the Software is recorded (cassette and/or diskette) or stored (ROM) is transferred to CUSTOMER, but not title to the Software.
- C. CUSTOMER may use Software on a multiuser or network system only if either, the Software is expressly labeled to be for use on a multiuser or network system, or one copy of this software is purchased for each node or terminal on which Software is to be used simultaneously.
- D. CUSTOMER shall not use, make, manufacture, or reproduce copies of Software except for use on **one** computer and as is specifically provided in this Software License. Customer is expressly prohibited from disassembling the Software.
- E. CUSTOMER is permitted to make additional copies of the Software **only** for backup or archival purposes or if additional copies are required in the operation of **one** computer with the Software, but only to the extent the Software allows a backup copy to be made. However, for TRSDOS Software, CUSTOMER is permitted to make a limited number of additional copies for CUSTOMER'S own use.
- F. CUSTOMER may resell or distribute unmodified copies of the Software provided CUSTOMER has purchased one copy of the Software for each one sold or distributed. The provisions of this Software License shall also be applicable to third parties receiving copies of the Software from CUSTOMER.
- G. All copyright notices shall be retained on all copies of the Software.

V. APPLICABILITY OF WARRANTY

- A. The terms and conditions of this Warranty are applicable as between RADIO SHACK and CUSTOMER to either a sale of the Equipment and/or Software License to CUSTOMER or to a transaction whereby Radio Shack sells or conveys such Equipment to a third party for lease to CUSTOMER.
- B. The limitations of liability and Warranty provisions herein shall inure to the benefit of RADIO SHACK, the author, owner and/or licensor of the Software and any manufacturer of the Equipment sold by Radio Shack.

VI. STATE LAW RIGHTS

The warranties granted herein give the **original** CUSTOMER specific legal rights, and the **original** CUSTOMER may have other rights which vary from state to state.

TANDY 3000

MS-DOS
REFERENCE
MANUAL

MS-DOS® Software: Copyright 1981, 1985
Microsoft Corporation. Licensed to Tandy
Corporation. All Rights Reserved.

MS-DOS is a registered trademark of Microsoft Corporation.

IBM is a registered trademark of International Business Machines
Corporation.

MS-DOS® Reference Manual.
Copyright 1985, 1986 Tandy Corporation.
All Rights Reserved.

Reproduction or use without express written permission from Tandy
Corporation of any portion of this manual is prohibited. While rea-
sonable efforts have been taken in the preparation of this manual to
assure its accuracy, Tandy Corporation assumes no liability result-
ing from any errors in or omissions from this manual, or from the
use of the information contained herein.

ABOUT YOUR MANUAL

MS-DOS is a powerful computer managing tool with numerous commands and options. This manual is intended as a reference to MS-DOS. It presents the general format for entering each command, describes the commands' options in detail, and gives examples of each command. If you are not familiar with MS-DOS, refer to your *MS-DOS Handbook*, which provides step-by-step instructions for using many of MS-DOS's basic commands and functions.

Part 1 describes the organization of directories and files and how to begin using MS-DOS commands and functions.

Part 2 tells you about such MS-DOS functions as wild cards, batch files, replaceable parameters and anonymous directory names. It includes a reference to all MS-DOS commands, with examples for using each command.

Part 3 explains MS-DOS editing functions, including command line editing and file editing using the MS-DOS line editor, EDLIN.

Parts 4 and 5 are for machine language programmers. Part 4 explains the use of the MS-DOS LINK program, used for linking machine language modules. Part 5 describes the DEBUG utility and its commands. Use DEBUG for changing machine language programs, locating *bugs*, and making corrections. Part 6 explains MS-DOS error messages.

The appendices provide a glossary, ASCII and Scan Code charts, ASCII character codes, and information about configuring your system for various peripheral devices.

CONTENTS

Part 1/Intro to MS-DOS

Chapter 1. How to Use MS-DOS	1
Entering a Command	1
Executing a Program	1
Editing and Special Keys	1
The Control Keys	3
Chapter 2. Organization of Information	7
The File System	7
The ROOT Directory	8
Filenames	8
Extensions	8
Examples of Filenames	9
Wild Cards	10
Pathnames	10
Device Names	11
Directories	12
Creating Directories	14
Deleting Directories	14
The Current Drive	15
The Current Directory	16
Home Directories	17
Anonymous Directory Names	18
Chapter 3. Redirecting Commands	21
Input And Output	21
Filters	22
Command Piping	22
Chapter 4. Batch Files	23
Executing Several Commands	23
The Batch File	23
Creating a Batch File	23
REM and PAUSE	24
Executing a Batch File	24
Summary of the Batch File Process	25
The Autoexec.bat File	25
Batch Files with Replaceable Parameters	26
Reminders about Batch Files	29

Part 2/MS-DOS Commands

Chapter 5. Introduction to MS-DOS Reference	31
Summary of MS-DOS Commands	32
How to Use the Command Reference	35
Command Structure and Syntax	36
Organization of MS-DOS Commands	37
The Use of Special Type	37
Synonymous Keywords	39
Chapter 6. MS-DOS Command Reference	41

Part 3/Editing

Chapter 7. Command Editing	187
The Template	187
Function Keys	187
Sample Session	188
Display and Change	188
Overtyping and Insert	189
Saving in the Template	190
Deleting	190
Chapter 8. File Editing	191
Creating a File	191
Inserting Text	192
Saving Your File	192
Editing an Existing File	193
Using the Special EDLIN Editing Keys	193
Example 1: Copy <i>Char</i>	194
Example 2: Copy to <i>Char</i>	194
Example 3: Copy All	195
Example 4: Delete <i>Char</i>	195
Example 5: Delete to <i>Char</i>	196
Example 6: Void Line	196
Example 7: Insert	197
Example 8: Replace Template	198
Example 9: Enter Line	199
Chapter 9. Introduction to EDLIN Reference	201
Summary of EDLIN Commands	201
Command Format and Rules	201
Command Parameters	203
Chapter 10. EDLIN Command Reference	205

Part 4/Linking

Chapter 11. Linking Object Modules	231
Basic Information	231
Starting and Using the Linker	233
Using the Prompt Method	233
Using the Command Line Method	235
Using the Response File Method	238
Giving Search Paths With Libraries	240
The Map File	241
The Temporary Disk File: VM.TMP	243
Using Linker Options	243
Pausing to Change Disks	244
Packing Executable Files	245
Producing a Public-Symbol Map	246
Copying Line Numbers to a Map File	246
Preserving Lowercase	247
Ignoring Default Libraries	248
Setting the Stack Size	248
Setting the Maximum Allocation Space	249
Setting a High Start Address	250
Allocating a Data Group	250
Removing Groups from a Program	251
Setting the Overlay Interrupt	251
Setting the Maximum Number of Segments	252
Using the DOS Segment Order	253
Chapter 12. How the Linker Works	255
Segments, Classes, and Groups	255
How the Linker Uses	
Segments, Classes, and Groups	256
Alignment of Segments	257
Canonical Frame Number	257
Order of Segments	258
Combined Segments	258
Groups	259
Fixups	259
Chapter 13. Using the Library Manager	261
Order of Operations	261
Starting and Using the Library Manager	262
Using the Prompt Method	262
Using the Command Line Method	263
Using the Response File Method	264
Command Characters	265

Part 5/Debugging

Chapter 14. Starting DEBUG..... 267

Chapter 15. Introduction to DEBUG Reference 269

Summary of DEBUG Commands..... 270

Command Parameters 271

Chapter 16. DEBUG Command Reference 275

Part 6/Messages

Chapter 17. Messages and Errors 307

The Messages 307

Disk and Device Errors 345

Part 7/Appendices

Appendix A. Glossary 347

Appendix B. ASCII and Scan Codes 353

 ASCII and Scan Code Special Handling 356

 ASCII Character Codes 356

Appendix C. Configuring Your System 365

CONFIG.SYS Commands 366

 BREAK 367

 BUFFERS 368

 COUNTRY 369

 DEVICE 370

 DRIVPARM 371

 FCBS 373

 FILES 374

 LASTDRIVE 375

 SHELL 376

Sample CONFIG.SYS File..... 377

Appendix D. Installable Device Drivers 379

ANSI.SYS 381

DRIVER.SYS 382

HDRIVE.SYS..... 384

LPDRVR.SYS..... 386

MLPART.SYS..... 391

MODEVM.SYS..... 392

SPOOLER.SYS 393

VDISK.SYS 394

Index 397

HOW TO USE MS-DOS

Entering a Command

You can enter a command whenever the screen displays the system prompt. The command can have a maximum of 125 characters, including any combination of upper- or lowercase letters. End each command by pressing **ENTER**. For example, type:

CLS **ENTER**

and MS-DOS executes the CLS command, which clears the screen and displays the system prompt.

If an error occurs while your computer is under the control of MS-DOS, the screen displays one of the error messages listed in Part 6.

Other error messages come from application programs. Check the instructions of those programs for explanations.

Executing a Program

You can also execute a program (such as the BASIC language application supplied on your system diskette) at the system prompt. If an entry is not a recognized command, MS-DOS compares it with the program names in the current directory. If it finds a match, MS-DOS loads and runs the program. Otherwise, the screen displays an error message. For example, type:

BASIC **ENTER**

to load BASIC.

Editing and Special Keys

Although you must be exact when entering commands, MS-DOS makes correcting mistyped entries easy. Rather than having to retype an entire command because of one error, you can use one of several special editing functions to correct the error.

You can also use these editing functions to change a previous command and *reuse* it.

Before entering a command. If you catch a mistake before you press `[ENTER]`, you can use one of the following keys to correct the mistake:

- `[←]` — moves the cursor one space to the left and erases the last character. You can then retype the character or characters.
- `[ESC]` — voids the command line and lets you start over.

After entering a command. If you type in your command without errors, pressing `[ENTER]` causes MS-DOS to execute the command. After execution, or if you made an error and pressed `[ENTER]`, you have the following options:

- Reexecute the same command. To do so, press `[F3]` `[ENTER]`.
- Edit the line and re-execute the command.
- Change the command or parameters and execute the new line.

To edit or change a line, you can use the MS-DOS special edit keys and functions. Among other things, you can:

- Insert letters or words
- Delete letters or words
- Copy to a specified character
- Delete to a specified character
- Replace a line
- Copy characters
- Void a line

The above editing abilities are also available in the MS-DOS file editing program, EDLIN. In addition to them, EDLIN offers the following:


















- Append lines
- List text
- Search text
- Transfer text

Information on editing both commands and files is presented in Part 3. You should study Part 3 thoroughly.

The Control Keys

In addition to the editing keys, MS-DOS provides a number of control keys. A control key affects a command or command line. The following chart shows all editing and control keys.

Note: When you type a control key sequence, press the second key while holding down the first.

Special Keys	Function
spacebar type a space	Moves the <i>cursor</i> (the blinking underline character displayed on the screen) one space to the right and adds a space to a line.
 display template	Displays the contents of the keyboard <i>template</i> (a storage place for the last line you typed) one character at a time. For more information on the template, see Part 3 of this manual.
 special commands	Lets you give complex commands to your computer by pressing only two or three keys. Hold  down while pressing another key.
 or   correct typing error	Moves the cursor left one character. Erases the character beneath the cursor. Use  to back up so that you can retype characters when you make mistakes.
  or   stop scroll	Stops the screen display from <i>scrolling</i> to let you view displays before they leave the screen.
  or   terminate process	Stops the execution of a program if the program uses MS-DOS functions. If the program does not access MS-DOS, the computer does not recognize this key sequence. Hold  down while pressing  . The computer might take a few moments before it recognizes the key sequence.

Special Keys	Function
[ENTER] process command carriage return	Starts the processing of a command line you typed. [ENTER] also causes a <i>carriage return</i> . (The cursor drops one line, and returns to the left margin.)
[CTRL] [P] or [CTRL] [PRTSC] print output	Causes all computer output to print on your printer (if it is connected and ready). Press the sequence again to stop the function.
[SHIFT] [PRTSC] print screen	Causes the current screen display to print (if your printer is connected and ready).
[CTRL] [J] end line	Causes the current line to end and performs a carriage return, but does not cause the line to be processed. You can continue to add to the command line before pressing [ENTER] .
[ESC] terminate line	Terminates the current line but does not process it. [ESC] clears the line buffer, outputs a backslash, a carriage return, and a line feed. [ESC] does not affect the template. Although the system prompt does not display, the system is ready for a command.
[F1] display character	Displays the next character in the template until you reach the end of the line. Serves the same function as [→] .
[F2] <i>char</i> copy to char	Copies all characters up to the specified character and displays them.
[F3] redisplay command	Redisplays on the screen the last line typed. You can cause the line to be processed again by pressing [ENTER] . To change the line, use [←] to erase characters. Then, type new characters.

Special Keys	Function
INS insert character	When you are editing a line, INS lets you insert characters. Press INS to begin inserting. Press INS again to end the insertion. Use the arrow keys to position the cursor where you wish the insertion to begin.
DEL delete character	When you are editing, DEL erases the character under the cursor. DEL erases a character each time you press it until you erase all characters to the right of the cursor.

ORGANIZATION OF INFORMATION

The MS-DOS system for organizing information, text, and programs on disk is efficient and easy to use. You can rename, copy, erase, filter, and edit files.

The File System

With MS-DOS, you can collect groups of files into *directories* — much as you might use a file cabinet drawer to collect all the file folders pertaining to a particular subject. You can then collect those directories, in turn, into larger directories.

It is important to remember this multilevel organization when working with MS-DOS files and directories. It lets you build downward, branching out as you go. In effect, you create an upside-down tree of files and directories.

Each user on the system can organize material into separate groups that are easy to locate and manage.

Here is a simplified diagram of a typical MS-DOS disk.

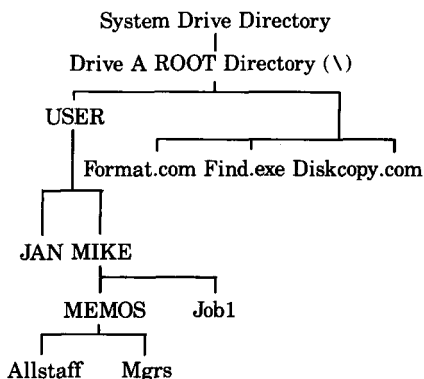


Fig. 2.1

Note: In this manual, directory names are in upper-case and filenames are in lower-case. This is to help you distinguish between directories and files. With MS-DOS, you can type all names in upper- or lower-case or in any combination of upper- and lower-case.

The ROOT Directory

Originally your MS-DOS system disk has only one directory, the ROOT directory, which contains all external command files. This directory is the root from which the rest of the disk's file system grows. The shorthand notation for a ROOT directory is a backward slash (\).

In the diagram (Figure 2.1), the user has added the USER directory to Drive A. (MS-DOS automatically creates the ROOT directory whenever you initialize a disk, using the FORMAT command.)

The USER directory contains two subdirectories, JAN and MIKE. The MIKE directory also contains a subdirectory, MEMOS, and a file, Job1. MEMOS contains two files, Allstaff and Mgrs.

Filenames

Each file has a name that can be a maximum of eight characters long. You can use letters, numbers, and some symbols in the name. Allowable characters are:

- Uppercase letters: (A-Z)
- Lowercase letters: (a-z)
- Decimal digits: (0-9)
- Symbols: \$ & # % ' () - @ ^ { } ' !

Do not include more than eight characters in a *filename*. MS-DOS truncates the filename to the first eight characters. For example, MS-DOS truncates both Accounts1 and Accounts2 to Accounts. Because two files (that are in the same directory) cannot have the same name, MS-DOS overwrites and destroys the old file.

Extensions

You can use an *extension* to provide additional information on a file. Always precede extensions with a period (.). Extensions can be a maximum of three characters long. Do not include more than three characters. If you do, MS-DOS truncates the extension to the first three characters.

Extensions such as .new, .irs, and .pay let you create files that have the same name (but different extensions), or they can help you divide files into categories.

You can also use an extension to indicate a file type, such as the following:

.bas	for	BASIC programs
.txt	for	ASCII text
.dat	for	Data files
.obj	for	Object code
.rel	for	Relocatable code
.src	for	Source code

If you add the extension .dat to a file named Invntory, the filespec becomes:

```
inventory.dat
```

Note: Once you include an extension in a filename, you must use it whenever you specify the file.

Examples of Filenames

Some legal filenames are:

```
rawdata2
REPORTS
X.x
project%.txt
PRDG1.bas
SAMFILE
2AR.dat
```

The examples below are illegal filenames:

max*min	(because * isn't a legal character for names)
.DATA	(because the period can be used only to separate the filename and extension)
open orders	(because a name can't contain a space and can only be a maximum of eight characters)

Wild Cards

MS-DOS lets you use these shorthand notations in filenames and extensions:

- ? The question mark indicates that any character can occupy that position.
- * The asterisk indicates that any character can occupy that position or the remaining positions in a filename or extension.

The following chart shows sample wild cards and the results you can expect when you use them.

You Type	MS-DOS Finds	Examples
test?run.exe	All files in one directory that begin with test, have any character next, followed by run and the extension .exe.	test1run.exe test3run.exe
test*.exe	The files above and all files in one directory that begin with test and have the extension .exe.	test.exe testall.exe test1.exe
oldfile.*	All files named Oldfile (in one directory), regardless of their extensions.	oldfile.bas oldfile.exe oldfile.txt

Pathnames

Because directories are organized in a multilevel fashion, MS-DOS needs exact directions (*pathnames*) to find files. A pathname specifies the drive a file is on, then lists the directories from the ROOT directory to the directory containing the file, and finally specifies the filename itself. Each pathname can have a maximum of 63 characters.

Refer to the previous MS-DOS disk diagram (Figure 2.1). To access the Job1 file on the disk in Drive A, tell MS-DOS the drive the file is on and the path from the ROOT directory to the file. Separate the *junctions* with backslashes. For instance, type:

```
TYPE A:\USER\mike\job1 ENTER
```


MS-DOS reads the pathname from left to right to determine that the file you want is in the MIKE directory on Drive A and that its name is Job1. It then lists the contents of Job1 to the screen.

MS-DOS's multilevel organization and pathname conventions help you access files quickly. You can give two files the same name as long as they are in separate directories. Under some circumstances, you can take a *shortcut* to a file or directory. For information on how to do this, see "Current Directory" later in this chapter.

Note: Some MS-DOS commands, such as MKDIR (make directory), require you to specify a directory instead of a file. In such a case, the pathname does not include a filename. Otherwise, it is the same as defined above.

Device Names

Each input/output device the system supports has a unique name. The device names are as follows:

- **AUX** (auxiliary) refers to the first RS232 serial port.
- **CON** (console) refers to the screen or keyboard.
- **PRN** (line printer1) refers to the first printer.
- **LPT1** (line printer 1) refers to the first printer.
- **LPT2** (line printer 2) refers to the second printer.
- **LPT3** (line printer 3) refers to the third printer.
- **COM1** (communications) refers to the first RS232 serial port.
- **COM2** (communications) refers to the second RS232 serial port.
- **NUL** (null) refers to a non-existent device.

Note: Avoid using the device names above as filenames. Doing so can cause unwanted results.

Directories

MS-DOS directories are collections of files. To understand how to create and use directories, look at the following example.

To create a file in the *current directory* (the directory you are using at a given time), using EDLIN, type:

```
EDLIN file1.txt 
```

The screen shows:

```
New file  
*
```

The asterisk indicates that EDLIN is ready for you to enter a command. To enter the insert mode, type:

```
I 
```

The screen shows the line number followed by a colon and an asterisk. EDLIN places each line you create into the text file until you type or to end the file. You do not type line numbers. EDLIN enters them automatically.

To create a file, type:

```
1:*This is my test file.   
2:*It shows the use of directories.   
3: 
```

After you type , EDLIN displays the asterisk to indicate it's ready for another command. To exit EDLIN and return to MS-DOS, type:

```
E 
```

The screen displays the system prompt again. Use the DIR command, which lists the files in a directory:

```
DIR 
```

The listing now indicates the existence of the new file, and displays the number of files in the current directory, as well as the number of free bytes on your diskette.

You can use the TYPE command to display the text stored in the file:

```
TYPE file1.tst 
```

The screen shows:

```
This is my test file.  
It shows the use of directories.
```

Use EDLIN to create two more text files.

```
EDLIN file2.tst 
```

The screen shows:

```
New file  
•
```

Type:

```
*I   
1:*This is my second file.   
2:*It shows the use of directories.   
3:*   
*E 
```

Create the third file:

```
EDLIN file3.tst 
```

The screen shows:

```
New file  
•
```

Type:

```
*I   
1:*This is my third file.   
2:*It shows the use of directories.   
3:*   
*E 
```

Now list the directory:

```
DIR 
```

The screen shows the directory of the *current disk* (the disk you are using), including the files you created:

FILE1	TST	58	8-24-84	9:07a
FILE2	TST	60	8-24-84	9:09a
FILE3	TST	59	8-24-84	9:09a

Creating Directories

The MKDIR command creates a new directory on any drive. To create a new directory called MYDIR as a subdirectory of the current directory, type:

```
MKDIR MYDIR 
```

MS-DOS automatically makes MYDIR a subdirectory of the current directory. You can check this by using DIR:

```
DIR 
```

The screen shows that MYDIR is a new subdirectory of the current directory.

To copy File1.tst to the new directory, use the COPY command as follows:

```
COPY file1.tst MYDIR\newfile1.tst 
```

Use DIR to see the name of the file in the new directory:

```
DIR MYDIR 
```

You can use MKDIR to create a subdirectory of MYDIR, a subdirectory of the new directory, and so on.

Deleting Directories

Before deleting a directory, first remove all files in that directory. Follow these steps:

1. Use the DIR command to see which files are in the directory.
2. Use the COPY command to copy any files you need to another directory.
3. Use the ERASE command to remove all files from the directory you are going to delete.

4. Use the RMDIR command to remove (delete) the directory.

If the directory you wish to delete contains subdirectories, follow the previously described procedure to delete the subdirectories. Then, delete the higher directory.

The Current Drive

The disk you are using at any given time is your *current disk* or *current drive*. Immediately after startup, MS-DOS places you in the ROOT directory of the Drive A disk if you boot using a floppy disk. In this case, Drive A is your current drive. If you boot using a hard disk, Drive C is your current drive.

Knowing about the current drive lets you take *shortcuts* when specifying pathnames. It also helps MS-DOS find what you want more quickly.

When specifying a file or directory that is *not* in the current drive, you must specify the entire pathname. However, when specifying a file or directory that *is* in the current drive, you need not include the drive specification in the pathname.

For example, suppose you are in A:\. To access C:\PAYROLL, specify the drive and the directory name. For example:

```
DIR C:\PAYROLL
```

To access A:\USER, specify only the directory name. For example:

```
DIR \USER
```

Changing the current drive. You can change the current drive quickly. To do so, enter the drive specification at the system prompt. For example, at A>, type:

```
C: 
```

The screen shows a new system prompt, C>, to indicate you are now in your hard disk Drive C.

The Current Directory

The directory you are using at any given time is your *current directory*. Immediately after startup, MS-DOS assigns the ROOT directory of Drive A (or Drive C with a hard disk) as your current directory.

Knowing about the current directory lets you take even more shortcuts when specifying pathnames. When specifying a file or directory that is higher than the current directory, and on the same disk, you must give a complete pathname (minus the drive specification). However, when specifying a file or directory that is within or below the current directory, and on the same disk, you can begin the pathname immediately below your current directory. The rest of the pathname is implied.

For example, suppose you are in the directory A:\USER\MIKE. To access the \USER directory on the same disk, give the complete pathname (minus the drive specification). For example:

```
DIR \USER
```

If you want to access the \USER\MIKE\MEMOS directory on the same disk, begin the pathname below the current directory. For example:

```
DIR MEMOS
```

Notice that you should not precede this pathname with a backslash. That is because the pathname \MEMOS specifies a directory that does not exist—the backslash means it would be an immediate subdirectory of the ROOT directory.

If you are still in \USER\MIKE, you can specify a file in that directory by giving only the filename. For example, if you type:

```
TYPE job1
```

The command implies the complete pathname \USER\MIKE\Job1.

Again, if you are still in \USER\MIKE and you specify MEMOS\Mgrs in a command, the command line implies the complete pathname \USER\MIKE\MEMOS\Mgrs.

When you enter commands, MS-DOS automatically expands pathnames as needed. For example, if you type:

```
COPY job1 MEMOS\newjob1 
```

from the USER\MIKE directory, MS-DOS interprets this as:

```
COPY \USER\MIKE\job1 \USER\MIKE\MEMOS\newjob1  

```

Changing the current directory. By using the CHDIR command, you can make any directory on the current drive the current directory. To do so, enter the CHDIR command followed by the pathname of the new current directory. For example, to change the current directory from A:\ to A:\USER\MIKE, type:

```
CHDIR \USER\MIKE 
```

To change to a directory on another drive, change drives before using the CHDIR command. For example, to change to a directory named PAYROLL on Drive C, first change your current drive to Drive C, by typing:

```
C: 
```

MS-DOS automatically puts you in the ROOT directory of Drive C. To change to \PAYROLL, type either of the following:

```
CHDIR PAYROLL  or
```

```
CHDIR \PAYROLL 
```

Home Directories

MS-DOS also *remembers* your previous current directory. If your current directory is C:\PAYROLL, and then you change it to A:\USER\MIKE, MS-DOS remembers that you were using C:\PAYROLL. For your convenience, it makes \PAYROLL the *home directory* of Drive C, until you change to another Drive C directory.

Using Home Directories. Whenever you specify a file or directory within or below C:\PAYROLL, you do not need to include the directory name PAYROLL. For example, suppose you are in Drive A. If you type:

```
TYPE C:allstaff 
```

MS-DOS assumes you mean C:\PAYROLL\Allstaff. Similarly, if you type:

```
DIR C: [ENTER]
```

MS-DOS assumes you mean DIR C:\PAYROLL. It shows a directory listing of all the files in the PAYROLL directory.

Knowing about home directories is particularly helpful when you copy files. Suppose you must copy several files from A:\USER\MIKE to C:\PAYROLL. Make C:\PAYROLL the home directory of Drive C. Then, make A:\USER\MIKE your current directory. Now you can copy each file, specifying only the filenames.

If you type:

```
COPY job1 C:newjob1 [ENTER]
```

MS-DOS interprets this as:

```
COPY A:\USER\MIKE\job1 C:\PAYROLL\newjob1 [ENTER]
```

Anonymous Directory Names

If you need to refer to your current directory or to a higher level directory, and do not know the full pathname — or if you simply want to save typing time — you can use *name substitutes*. For example:

- The name “.” refers to the current directory.
- The name “..” refers to the *parent* of the current directory (the next-higher directory in the path).
- The name “..\..” refers to the directory two levels up.

You can use an anonymous name in place of a pathname, or as the first directory name in a pathname. For example,

```
DIR . [ENTER]
```

lists filenames in the current directory.

```
ERASE ..\taxes84 [ENTER]
```

deletes the Taxes84 file from the current directory's parent directory.

Substitute names can refer to either the current directory or the home directory of another drive, depending on whether or not you include a drive specification. For example:

```
ERASE C:..\taxes84 
```

deletes the Taxes84 file from the parent directory of the home directory of Drive C.

If you want to copy several files from A:\USER\MIKE to C:\PAYROLL, make A:\USER\MIKE the current directory of Drive A, and use CHDIR (CD) to change the Drive C directory to C:\PAYROLL. Then type:

```
C:   
COPY A:job1 
```

MS-DOS interprets this as:

```
COPY A:\USER\MIKE\job1 C:\PAYROLL\job1 
```


REDIRECTING COMMANDS

MS-DOS assumes that command *input* comes from the keyboard and *output* goes to the screen. However, you can *redirect* input so it comes from a file and output so it goes to a file or a printer. In addition, you can create *pipes* that let one command's output become another's input.

Input and Output

To redirect input so it comes from a file, use a less-than sign (<) in your command. For example:

```
SORT <names 
```

sorts the information in the file Names and displays the sorted output.

To redirect output so it goes to a file, use a greater-than sign (>) in your command. For example:

```
DIR >myfiles 
```

sends a directory listing to the file Myfiles in the current directory. If Myfiles does not already exist, MS-DOS creates it and stores the listing in it. If Myfiles does exist, MS-DOS overwrites the old information with the new information.

Append. To add your output to the end of an existing file, use two greater-than signs. For example:

```
DIR >>myfiles 
```

appends your directory listing to the file Myfiles in the current directory. If Myfiles does not exist, MS-DOS creates it. The following command redirects both the input and the output.

```
SORT <names >list1 
```

This command sorts the information in the file Names, and sends the sorted output to the file List1 in the current directory.

Filters

A *filter* is a command that *operates* on data in some way before outputting it—usually to the screen or a file. The MS-DOS filters and functions are:

- **FIND** — Searches for occurrences of a requested string of text.
- **MORE** — Causes the screen output to be displayed one screen at a time. To view the next screen, press any key.
- **SORT** — Sorts text from A-Z or from Z-A (reverse sort).

Note: By combining commands and filters, you can replace several commands with a few filters.

Command Piping

Piping lets you give more than one command at a time to the system. It does this by making one command's output another command's input.

To pipe commands, place them on one command line and divide them with the *pipe separator*, the vertical bar (`|`). All output generated by the command to the left of the bar becomes input for the command to the right.

For example, you can sort your directory listing. This command:

```
DIR | SORT [ENTER]
```

displays an alphabetically sorted listing of the current directory. This command:

```
DIR | SORT >direc.fil [ENTER]
```

sends the same listing to the file `Direc.fil` in the current directory. If the file does not exist, MS-DOS creates it. This command:

BATCH FILES

Executing Several Commands

Some tasks require two or more commands. For example, when preparing a disk for information storage, you must format the disk so you can write to it (FORMAT command). In addition, it is a good habit to immediately check the directory of the new disk for errors (CHKDSK command).

The Batch File

You can put a command sequence into a special file called a *batch* file. Then, you can execute the entire sequence by entering the name of the batch file.

When creating a batch file, you can specify the complete path-name or a filespec. In either case, give the file the extension .bat. When you execute the file, however, enter the filename without the extension.

Below is a description of how to use the COPY command to create a batch file. (You can also use EDLIN. See Part 3.)

Creating a Batch File

To create a file to perform a command sequence of FORMAT and CHKDSK, type this at the system prompt:

```
COPY con prepdisk.bat 
```

This command tells MS-DOS to copy the information entered from the keyboard (console) into a batch file called Prepdisk.bat. Because you do not specify otherwise, MS-DOS creates the file in the current directory.

MS-DOS displays the cursor. Now, you can type the commands to be included in the file:

```
REM This prepares and checks new disks   
REM It is called Prepdisk.bat   
FORMAT   
PAUSE   
CHKDSK 
```

To save the commands in a file, type **F6** **ENTER** or **CTRL** **Z** **ENTER**. MS-DOS displays:

```
1 File(s) Copied
```

Type **DIR** **ENTER** to verify that the file is created.

REM and PAUSE

You use **REM** and **PAUSE** only in batch files. The **REM** command lets you include remarks in your batch file. Remarks do not affect the operation of the command. The remarks in the **Prepdisk.bat** file are to remind you what the file does. If you forget, you can type:

```
TYPE prepdisk.bat ENTER
```

and MS-DOS displays the contents of the file, including the remarks.

The **PAUSE** command lets you control how much of the file you want to execute. When it reaches a **PAUSE** command, MS-DOS pauses and displays the message:

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

You can continue execution by pressing any key, or you can halt it by pressing **CTRL** **C**.

Executing a Batch File

To execute a batch file, type the name of the file and press **ENTER**. If you are not in the same directory as the batch file, use the file's complete pathname. In either case, omit the extension. For example, to execute **Prepdisk.bat**, type this at the system prompt:

```
prepdisk ENTER
```

Execution proceeds as if you entered each command line at the keyboard.

A word of advice about batch files. If you are not familiar with MS-DOS, it is easiest to create all batch files in the ROOT directory of your disk. Later, when you better understand MS-DOS, you might want to create batch files elsewhere. You can do this by specifying a complete pathname for the file. If you do so, however, keep in mind the following:

- If you are not in the same directory that contains the batch file to be executed, type the full pathname.
- MS-DOS searches only the current directory for external commands. If you use any external commands in your batch file, you need to do one of the following:
 - Use PATH to tell MS-DOS to search for external commands in the directory that contains the commands, or
 - Use COPY to move your external commands to the directory that contains your batch file, or
 - Use CHDIR within your batch file to move from one directory to another as necessary.

Summary of the Batch File Process

1. Create the file by typing:

`COPY con pathname.bat ENTER`

2. Enter the command lines.

3. Save the file by typing either of the following:

F6 ENTER or
CTRL Z ENTER

4. When you are in the directory that contains the file, execute the file by typing:

`filename ENTER`

The Autoexec.bat File

An Autoexec.bat file executes programs or commands automatically when you start MS-DOS. By creating an appropriate batch file, you can run an application program or execute a series of commands each time you start the system.

When you start MS-DOS, the command processor searches the MS-DOS disk for a file called Autoexec.bat. If MS-DOS finds the file, it immediately executes it.

Note: If you use an Autoexec.bat file, MS-DOS prompts for a current date and time only if you include the DATE and TIME commands in the file. Because MS-DOS uses this information to keep the directory current, you should always include these commands.

Creating an Autoexec.bat file. The Autoexec.bat file must be created in the ROOT directory of your MS-DOS disk. (You must be in that directory when you create the file.) You create it the same way as any other batch file, except that you name it Autoexec.bat.

To create and save a file that loads BASIC and runs a BASIC program named MENU each time you start MS-DOS, type:

```
COPY con autoexec.bat   
DATE   
TIME   
BASIC MENU   
 
```

Batch Files With Replaceable Parameters

When creating a batch file, you can include *replaceable* (dummy) parameters. Doing so lets you use a different set of data each time you run the file. For example, the Prepdisk.bat file shown earlier is changed in the following listing to include the dummy parameter %1.

```
COPY con prepdisk.bat   
REM This prepares and checks new disks   
REM in the drive you specify   
REM It is called Prepdisk.bat   
FORMAT %1   
CHKDSK %1   
 
```

To use this file to format the Drive A disk, enter the batch file name and the drive specification to replace %1, as shown here:

```
prepdisk A: 
```


The next two sections tell more about creating and executing batch files that have dummy parameters.

Creating a Batch File With Replaceable Parameters. The dummy parameters are %0 through %9. MS-DOS always replaces the %0 parameter with the filename of the batch file, unless you use the SHIFT command. It replaces the other dummy parameters, sequentially, with the parameters you specify when you execute the batch file. (See SHIFT in Part 2 if you wish to specify more than 10 dummy parameters.)

The parameters can be pathnames, drive specifications, numeric values, or almost anything else. For example, if you type:

```
COPY con myfile.bat   
COPY %1.mac %2.mac   
TYPE %2.mac   
TYPE %0.bat   
 
```

MS-DOS creates the batch file Myfile.bat in the current directory and stores the next three lines in that file.

When you execute the file, MS-DOS replaces the parameters %1 and %2, sequentially, with the pathnames you supply.

Executing a Batch File With Replaceable Parameters. To execute a batch file that has replaceable parameters, enter the batch filespec (without its extension), followed by the parameters to replace the dummy parameters. For example, to execute Myfile.bat, type:

```
myfile A:prog1 C:prog2 
```

MS-DOS substitutes Myfile for %0, A:Prog1 for %1, and C:Prog2 for %2.

The result is the same as if you enter each command with its parameters, as follows:

```
COPY A:prog1.mac C:prog2.mac   
TYPE C:prog2.mac   
TYPE myfile.bat 
```

The COPY command copies the contents of one file to another file. The TYPE command displays the contents of a file.

Sample Use Of Replaceable Parameters. Replaceable parameters can be useful with application programs. Suppose you have a program that creates a client mailing list and saves the list to the Mail.dat file in the ROOT directory of your system disk.

The information in Mail.dat might look like this:

```
Tom Cleo 2 8th St. Lincoln NE 68502
Ann King 1 9th Av. Rapid City SD 57001
Sam Beck 4 6th St. Ft. Worth TX 76133
```

As you can see, the information is in random order. Using a batch file with replaceable parameters, however, you can organize the information in any way that is most convenient at any given time. One time, you might organize it according to zip code; another time, according to last name; and so on. To create such a batch file, type the following:

```
COPY con mailsort.bat 
COPY mail.dat %1.dat 
TYPE %1.dat 
SORT %2 <%1.dat >%3.dat 
TYPE %3.dat 
F6 
```

Execute the batch file by typing:

```
mailsort newmail /+5 sort 
```

The result is the same as if you enter each command with its parameters, as follows:

```
COPY mail.dat newmail.dat 
TYPE newmail.dat 
SORT/+5 <newmail.dat >sort.dat 
TYPE sort.dat 
```

MS-DOS copies the information from Mail.dat into Newmail.dat, and then displays the contents. Starting at Column 5, it alphabetically sorts the file Newmail.dat. Then, it copies sorted data to the file Sort.dat, and displays Sort.dat.

Reminders About Batch Files

The following list summarizes information you should know before you execute a batch process with MS-DOS.

- Do not enter the filename Batch (unless the name of the file you want to execute is Batch.bat).
- To execute a batch file, enter the filename without the extension.
- If you press **CTRL C** while in the batch mode, this prompt appears:

Terminate batch job (Y/N)?

If you press **Y**, MS-DOS ignores the rest of the commands in the batch file. The screen displays the system prompt.

If you press **N**, the current command ends. Batch processing continues with the next command in the file.

- Removing the disk that contains an executing batch file causes an error.
- Immediately upon executing one batch file, you can call another. To do so, put the second file's name (without its extension) as the last command in the first file.
- If, in a batch file, you want to refer to a file that has a name containing a percent sign, you must include a second percent sign. A batch file normally interprets the percent sign as a replaceable parameter. A second percent sign indicates that this is not the case. If the file to which you are referring is abc%.exe, the batch file command might be this:

```
COPY A:\USER\abc%%.exe C:\USER
```


INTRODUCTION TO MS-DOS REFERENCE

There are two types of MS-DOS commands, *internal* and *external*. Internal commands are the simpler, more commonly used commands. When you list a directory, you cannot see these commands. When you enter them, they execute immediately.

External commands reside on disks as program files. Therefore, MS-DOS must read them from disk before it can execute them. If the disk containing the command is not in the drive, the system cannot find and execute the command.

Note: Unless MS-DOS knows in which directory or drive to search for external commands, it cannot execute them. (See the PATH command.)

Any filename that has an extension of .com, .exe, or .bat is considered an external command. For example, programs such as FORMAT.COM and DISKCOPY.COM are external commands. You can create external commands and add them to the system. Programs that you create with most languages (including assembly language) are .exe (executable) files.

When you enter an external command, do not include its filename extension.

Summary of MS-DOS Commands

COMMAND	PURPOSE
APPEND	Sets a data file path
ASSIGN	Reassigns drive names
ATTRIB	Sets or displays a file's read-only or archive attributes
BACKUP	Backs up files from one disk to another
BREAK	Turns the <input type="checkbox"/> CTRL <input type="checkbox"/> C check on or off
CHDIR	Changes current or home directories
CHKDSK	Checks MS-DOS diskettes
CLS	Clears the video screen
COMMAND	Starts a new command processor
COPY	Copies, appends, or combines files
CTTY	Changes the input/output device
DATE	Enters or changes the system date
DEL	Deletes files from the specified directory
DIR	Displays files in the specified directory
DISKCOMP	Compares two diskettes
DISKCOPY	Makes copies of floppy diskettes
DISKTYPE	Displays information about the size and capacity of a disk
ECHO	Controls the display of lines in batch files
ERASE	Deletes the specified file or files
EXE2BIN	Converts .exe files to binary format
EXIT	Exits from commands to the previous level
FC	Compares the contents of two files
FDISK	Partitions a hard disk
FIND	Searches for the specified text

Summary of MS-DOS Commands (Continued)

COMMAND	PURPOSE
FMAT2000	Formats a standard diskette for 720K bytes in a high-capacity drive
FOR	Executes the same command for several items with one command line
FORMAT	Prepares a disk for system use
GOTO	Transfers execution to the selected routine in a batch file
GRAFTABL	Installs alternate ASCII characters into memory
GRAPHICS	Enables you to reproduce a graphics screen in color on a Tandy CGP-220 or in shades of gray on other printers
HSECT	Formats track and sector information on a hard disk
IF	Allows conditional execution of commands in batch files
JOIN	Joins a disk drive to a pathname
KEYBFR, KEYBGR, KEYBIT, KEYBSP, KEYBUK	Replaces the keyboard BIOS with an international program from France, Germany, Italy, Spain, or the United Kingdom
LABEL	Creates, changes, or deletes volume labels
LF	Suppresses line feed after a carriage return
MKDIR	Creates a new directory
MLFORMAT	Formats non-bootable (DOS2) hard disk partitions
MLPART	Creates non-bootable (DOS2) partitions

Summary of MS-DOS Commands (Continued)

COMMAND	PURPOSE
MODE	Sets video, printer, and communication parameters, and the CPU speed
MORE	Displays one screen of information at a time
PATCH	Lets you make minor modifications to a disk file
PATH	Specifies the path to the external commands
PAUSE	Suspends batch execution; displays the specified message
PRINT	Puts files in the print queue for background printing
PROMPT	Creates a new system prompt
RECOVER	Recovers bad sectors on a disk
REM	Allows comments in a batch file
REN	Changes a file's name (renames the file)
REPLACE	Updates previous versions of files
RESTORE	Restores files previously backed up with BACKUP
RMDIR (RD)	Deletes (removes) a specified directory
SELECT	Selects country-dependent information
SET	Sets one string value to another in the environment or displays the SET values
SETUP	Initializes the system configuration information in CMOS RAM
SHARE	Installs file sharing and locking for active networking
SHIFT	Shifts the definitions of replaceable parameters in batch files
SHIPTRAK	Parks hard disk heads for transportation

Summary of MS-DOS Commands (Continued)

COMMAND	PURPOSE
SORT	Sorts input from the keyboard or a file
SPOOLER	Operates the printer spooler
SUBST	Substitutes a virtual drive name for a pathname
SYS	Transfers the MS-DOS system files from one disk to another
TIME	Displays or sets the system time
TREE	Displays all disk directories and files
TYPE	Displays the contents of the specified file
VER	Displays the MS-DOS version number
VERIFY	Verifies that files are intact
VOL	Displays the volume label of the specified disk
XCOPY	Copies files and directories from one disk to another

How to Use the Command Reference

Command lines can be divided into two parts, the command name and the command parameters. Some parameters are required; others are optional. If you omit an optional parameter, the system provides a *default* parameter. For example, the system defaults to the current drive whenever you omit the drive from the pathname.

Required and optional parameters and default values vary with different commands. Here is an example of how to interpret a command line:

COPY *source pathname* [*target pathname*] [/A]/[B]/[V]

- **COPY** is the command name you type to call up the COPY function. You can enter a command name in either upper- or lowercase. For your convenience, command names are presented in uppercase in this manual.

- *source pathname* is a required parameter. Parameters not enclosed in brackets are essential to the operation of the command; so, you must include them in the command line. In the example, *source pathname* is the name of the file you wish to copy. Example: COPY A:Myfile.
- *target pathname* is an optional parameter. Include descriptions or values for parameters enclosed in brackets **only** if you want to change the default description or value. In the example, if you omit *target pathname*, MS-DOS gives the new copy the same filename as the original file.
- /A, /B, /V are optional *switches*. Again, include them in the command line only if you wish to change the default settings of these switches. In the example, the switches are used to copy files in ASCII format and to add or delete EOF characters from the files.

Command Structure And Syntax

- The default system prompt is the current drive designation followed by a greater-than sign. For example, A> tells you that Drive A is your current drive and that MS-DOS is ready to accept a command. You can use the PROMPT command to change the default prompt.
- Commands are usually followed by one or more parameters.
- You can enter parameters in uppercase, lowercase, or any combination.
- You must separate commands and parameters with delimiters. The space and comma are the easiest to use. Examples:

```
DEL MYFILE.OLD NEWFILE.TXT  
rename,afile bfile
```
- You can also use the semicolon, equal sign, or tab as delimiters. This manual uses a space.
- *Source* specifies the disk from which you transfer information. *Target* specifies the disk to which you transfer information. Some error messages refer to the *destination*. This is the same as the *target*.

- When typing commands, you can use the MS-DOS editing and function keys. (See Chapters 2 and 7 for editing information.)
- Commands execute only after you press **ENTER**.
- Pressing **CTRL C** or **CTRL BREAK** halts a command when it is running.
- When a command produces more output than the screen can display, the display begins to scroll up. To stop scrolling, press **CTRL S**. To resume, press the space bar.

Organization of MS-DOS Commands

The explanation of each MS-DOS command in Part 2 consists of several items, which are arranged in the following order:

- The command name and type (external or internal).
- The command *syntax*: a guide for entering the command.
- Parameters: a list of all the command's parameters and their functions.
- Notes and Suggestions: information and suggestions about using the command.
- Examples: samples of how to use the command.

Note: Part 6 contains a complete listing of the error messages associated with the MS-DOS commands.

The Use of Special Type

KEYBOARD CHARACTER: indicates a key you press.

lowercase italics: represent words, letters, characters, or values that you supply.

UPPERCASE letters: indicate *keywords* (command calls) that you must type. You can type the keywords in any combination of upper- and lowercase letters. MS-DOS interprets them as uppercase.

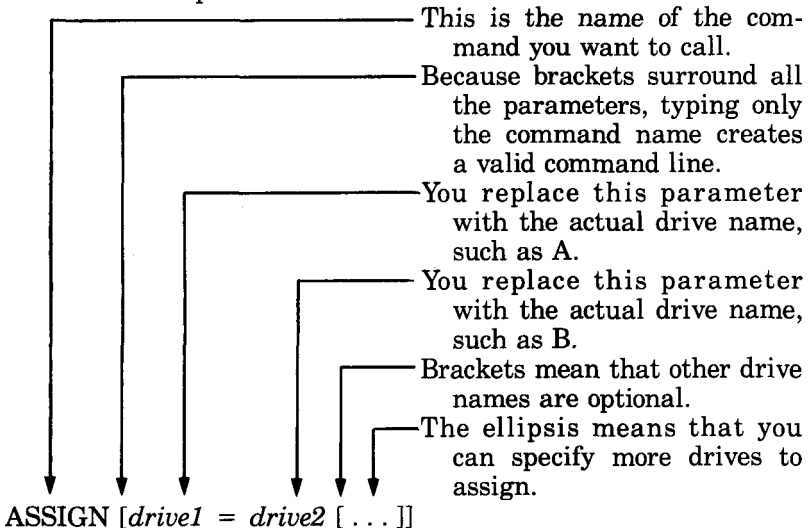
[] (square brackets): indicate optional command parameters. When using the parameters, do not include the brackets.

... (ellipsis): indicates that you can repeat a parameter as many times as you wish.

! (vertical bar): indicates an either/or situation.

Note: Type all other punctuation exactly as shown in the syntax line of the command.

The following demonstrates a command syntax line and the functions of its parts:



This command syntax tells you that there are numerous options available. For instance, the following examples are but some of the valid format lines for ASSIGN.

```
ASSIGN [ENTER]
assign A = B [ENTER]
ASSIGN a = b [ENTER]
ASSIGN A = B C = D [ENTER]
```

After you issue an ASSIGN command such as these, MS-DOS assumes that whenever it sees *drive1* (for example, Drive A) in a command line you want it to access *drive2* (for example, Drive B).

Note: ASSIGN is the only MS-DOS command that does not require a colon following a drive letter.

Synonymous Keywords

Some commands give you a choice of keywords to accomplish the same task. For example, you can type either DEL or ERASE when deleting files. You can also shorten MKDIR to MD, CHDIR to CD, and RMDIR to RD.

MS-DOS COMMAND REFERENCE

APPEND

External

APPEND [;] [*pathname* [;*pathname*]...]

Sets a data file path, which tells MS-DOS the drives and directories in which to search for data files; displays the current path.

Parameters

pathname is the directory you want MS-DOS to search. You can specify more than one path by separating the pathnames with semicolons (;).

Notes and Suggestions

- If you omit all options, MS-DOS displays the current data path.
- To set the NUL data path so that MS-DOS searches only the current directory, type:

APPEND ;

- APPEND searches the data path for all files, regardless of their file extensions, only with the following system calls:

Code	Function
0FH	Open file (FCB)
23H	Get file size
3DH	Open handles

- You can use APPEND across a network to locate remote data files.

Example

APPEND B:\DATADIR

searches the \DATADIR directory on Drive B for all data files.

ASSIGN

External

ASSIGN [*drive1* = *drive2* . . .]

Reassigns a drive letter to a different drive.

Parameters

drive1 is the drive to which reads and writes are currently sent.

drive2 is the drive to which you want reads and writes sent whenever you specify the letter *drive1*.

Neither parameter requires a colon after the drive letter.

If you omit both parameters, ASSIGN reassigns all drive letters to their normal drives.

Notes and Suggestions

- ASSIGN lets you use application programs on drives other than those for which they were specifically designed. For example, you can use a program that is designed for Drives A and B on Drive C.
- Do not use ASSIGN with the BACKUP or PRINT command or during normal use of MS-DOS. BACKUP and PRINT require true device type information, which ASSIGN hides.
- The FORMAT and DISKCOPY command ignore any drive letter reassignments.
- The equal sign (=) is optional.

Example

ASSIGN A=C B=C

assigns the drive letters A and B to Drive C so that references to Drives A and B access Drive C. (Note: References to Drive C continue to access Drive C, also.)

ATTRIB

External

ATTRIB [+R | -R] [+A | -A] *pathname*

Sets or resets the read-only and archive attributes of a file; displays the attributes of a file.

Parameters

- | | |
|----|-------------------------------|
| +R | Sets the read-only mode. |
| -R | disables the read-only mode. |
| +A | sets the archive attribute. |
| -A | clears the archive attribute. |

pathname specifies the file for which you want to set or display attributes. If you specify only the *pathname*, ATTRIB displays the file's attributes.

Notes and Suggestions

- If an application opens a file with read and write permission, you can use ATTRIB to force a read-only mode to allow file sharing over a network.
- The BACKUP, RESTORE, and XCOPY commands use the archive attribute to control selective backups, restores, and copies on modified files. You can use the +A and -A options to change a file's archive status so that you have further control over which files you back up or copy using the /M switch.
- You can use the wild card *.* to display the attributes of all files on a specific drive.

Example

```
ATTRIB +R myfile.txt ENTER
```

sets the attribute of Myfile.txt to read-only.

BACKUP

External

BACKUP [*pathname*] [*drive*] [/S] [/M] [/A] [/P]
[/D:*mm-dd-yy*] [/T:*hh:mmx*] [/L:*filename*]

Backs up one or more files from one disk to another formatted disk. BACKUP can copy between disks of different media, for example from hard disk drives to floppy disk drives. It can also copy from one floppy diskette to another, even if the diskettes have a different number of sides and sectors.

Parameters

<i>pathname</i>	specifies the files to back up. It can be an entire drive, a directory name, or a filename.
<i>drive</i>	is the drive to receive the files. If the target is a floppy disk drive, BACKUP places the files in the ROOT directory. If it is a hard disk drive, BACKUP places the files in a subdirectory called BACKUP.
/S	copies all files in the specified directory and in the directories below it.
/M	copies only those files modified since the last backup.
/A	adds the files to be backed up to those already on the target disk, instead of erasing the existing files.
/P	packs as many files as possible on each disk, creating a subdirectory on the target if necessary. (Caution: You might lose IBM® BACKUP/RESTORE compatibility if you use this switch.)
/D: <i>mm-dd-yy</i>	copies only those files created on or after the specified date.
/T: <i>hh:mmx</i>	backs up only files modified at or after the specified time. The time must be less than 13:00. X is either A (for a.m.) or P (for p.m.).

/L:filename

creates a backup log entry in the specified file or—if you omit *filename*—in a file called Backup.log in the ROOT directory of the files being backed up. (Caution: You might lose IBM® BACKUP/RESTORE compatibility if you use this switch.)

The first line of the backup log file contains the backup date and times. Each subsequent line corresponds to one of the files backed up. The line contains the filename and the number of the backup disk containing the file. This information is useful when you restore a particular file. By looking at the Backup.log file, you know which disk to specify for RESTORE. On subsequent BACKUP operations, because Backup.log already exists, the current entry is appended to the file.

Notes and Suggestions

- Unless you specify otherwise, BACKUP erases the old files on the target disk before storing the new files there.
- If more than one target disk is required, BACKUP prompts you to exchange disks when the current disk is full.
- BACKUP displays the name of each file as it backs up the file. Label and number each backup disk consecutively to help you later restore the files properly.
- If you are sharing files, you can back up only those you have access to.
- Do not use the BACKUP command if the target drive is assigned, joined, or substituted. If you do, you might not be able to restore the files.

Examples

```
BACKUP C: A: /S 
```

copies the files in the current hard disk directory to a diskette in Drive A. This operation erases any existing files on the Drive A diskette.

```
BACKUP C:myfile A: /A 
```

copies the file Myfile from the current directory of Drive C to the floppy diskette in Drive A. This operation adds the new file to the files already on the Drive A diskette, without erasing any.

```
BACKUP *.* A: /P 
```

copies all files in the current directory to the diskette in Drive A. The files are packed on the diskette as economically as possible. Any existing files on the diskette in Drive A are erased.

```
BACKUP C:STORE1 A: /L /M 
```

copies to Drive A only those files from the STORE1 directory on Drive C that have been updated (changed) since the last BACKUP operation. Creates a Backup.log file in the ROOT directory.

BREAK

Internal

BREAK [ON|OFF]

Turns the **CTRL** **BREAK** check on or off or displays the current setting. The **BREAK** command affects only application programs. It has no effect at the MS-DOS command level or to BASIC programs. (**CTRL** **BREAK** and **CTRL** **C** serve identical purposes. Wherever this manual refers to **CTRL** **BREAK**, the reference applies to **CTRL** **C**, too.)

Parameters

- ON** tells MS-DOS to check for **CTRL** **BREAK** from the keyboard whenever an application program makes any type of MS-DOS function call.
- OFF** tells MS-DOS to check for a **CTRL** **BREAK** only when a screen, keyboard, printer, or serial port function call is made.

If you omit both **ON** and **OFF**, MS-DOS displays the current setting of **BREAK**.

Examples

BREAK OFF **ENTER**

turns off the MS-DOS **CTRL** **BREAK** check. Use this command before running an application program that uses the **CTRL** **BREAK** function keys. Pressing **CTRL** **BREAK** affects your program instead of the operating system.

BREAK ON **ENTER**

turns on the **CTRL** **BREAK** check. Use this command after you finish running your application program and are planning to use MS-DOS.

BREAK **ENTER**

displays the current setting of **BREAK**.

CHDIR (Change Directory)

Internal

CHDIR [*pathname*]

CD [*pathname*]

Changes the current, or home, directory of the specified drive to the directory specified by *pathname*. CHDIR also verifies the current directory.

Parameters

pathname

specifies the directory that is to be the current directory. *pathname* must be another directory on the current disk. If you are changing the home directory of a disk other than the current disk, you must specify the drive that contains the directory, and *pathname* must be a directory on that disk.

If you omit *pathname*, MS-DOS displays the *pathname* of your current directory. This lets you verify a directory change or the name of a directory should you forget.

Notes and Suggestions

- Changing directories before entering commands can save you considerable time. When executing an application program, you are likely to store your information in several data files in the same directory. Therefore, it can be convenient to make that directory your current directory.

Suppose you use a mailing list program to keep track of magazine subscriptions. You might create three data files: (1) sorted by your customers' last names, (2) sorted by zip code, and (3) sorted by subscription expiration date.

You can store all three files in a directory called \MAGMAIL. When running the program, make \MAGMAIL your current directory. Then, you can quickly access and transfer data between files.

- **Hard Disk Users:** Because you are likely to store several application programs on your hard disk, it is useful to put each in a separate directory. (See MKDIR.) For example, you might want to store an accounts receivable program in a directory called \AR. When you are ready to use the program, use CHDIR to make \AR your current directory.
- To change your current directory to a directory on another disk, first make that disk your current disk. Now, use the CHDIR command to change directories. The following examples illustrate these concepts.

Examples

```
B: [ENTER]  
CHDIR \USER [ENTER]
```

changes the current drive to Drive B; then, changes the current directory to \USER.

```
CHDIR .. [ENTER]
```

puts you in (changes your current directory to) the parent directory of your current directory.

```
CHDIR \ [ENTER]
```

puts you in the root directory of the current disk.

```
CHDIR \BIN\USER [ENTER]
```

puts you in the directory \BIN\USER on the current disk.

```
CHDIR [ENTER]
```

displays the pathname of your current directory. If, for example, you are in the directory \BIN\USER on the disk in Drive B, MS-DOS displays B:\BIN\USER.

```
CHDIR B:\USER 
```

changes the home directory of Drive B to USER.

```
CHDIR \USER\LETTERS 
```

```
CHDIR B:\USER\MEMOS 
```

(1) changes the current directory on Drive A from the ROOT directory to \USER\LETTERS and (2) changes the home directory of Drive B from the root directory to \USER\MEMOS. Now, instead of typing:

```
COPY B:\USER\MEMOS\filename  
A:\USER\LETTERS\filename 
```

for every file you copy between the two directories, you need only type:

```
COPY B:filename filename 
```

for each file you copy.

CHKDSK (Check Disk)

External

CHKDSK [*pathname*] [/F] [/V]

Checks the directory of the MS-DOS disk in the current or specified drive for errors. You should run CHKDSK occasionally on each disk. CHKDSK does not prompt you to insert the disk; it assumes the disk is in the drive. After checking the directory, CHKDSK displays the proper error messages, if any, and then gives a status report.

Parameters

<i>pathname</i>	specifies either an entire drive or an individual file to be checked. If you specify a file, CHKDSK displays information about both the drive and the file.
/F	causes CHKDSK to fix any errors it can and to update the disk.
/V	causes CHKDSK to display messages while it is running.

Notes and Suggestions

- Here is a sample report produced by CHKDSK. The numbers vary on different disks.

```
Volume DDSDISK    created Jan 12, 1986
                  3:21 p

    362496 bytes total disk space
    38912 bytes in 2 hidden files
     1024 bytes in 2 directories
   282624 bytes in 8 user files
    39936 bytes available on disk

    524288 bytes total memory
    487248 bytes free
```

The report includes 2 hidden files. These are the system files. You cannot see these files when you use the DIR command.

- If you want to save the status report for future use, you can redirect CHKDSK's output to a file. To do this, append *>pathname* to the end of the command line. For example, to check Drive A and redirect output to a file called Errors in the \USER\TOM subdirectory on Drive B, type:

```
CHKDSK A:>B:\USER\TOM\errors ENTER
```

Do not use the /F switch if you are redirecting output.

- You cannot use the CHKDSK command over a network.
- The following are messages that CHKDSK displays for your information. They are not errors:

(.) (...) does not exist
The . or ... directory entry is invalid.

(filename) contains
non-contiguous blocks
The file or files you named are not written
contiguously on the disk.

All specified file(s) are contiguous
The file or files you named are written contiguously on the disk.

- The following are errors that MS-DOS corrects automatically if you use the /F switch with CHKDSK. For more information on these errors, see Part 6.

Allocation error, size adjusted

Entry has a bad attribute (or size
or link)

Errors found, F parameter not
specified
Corrections will not be
written to disk

First cluster number is invalid
Entry truncated

x lost clusters found in y chains
Convert lost chains to files (Y/N)?

- The following are errors that MS-DOS cannot correct automatically, even if you use /F. For information on how to correct the errors, see Part 6.

(filename) is cross-linked on cluster

Cannot Chdir to (filename)

Tree past this point not processed

Cannot Chdir to root

Processing cannot continue

Cannot recover . entry, processing continued

Cannot recover .. entry

Directory is totally empty, no . or ..

Disk error reading FAT

Disk error writing FAT

Incorrect DOS version

Insufficient memory

Processing cannot continue

Insufficient room in root directory

Erase files in root and repeat Chkdsk

Invalid drive specification

Invalid parameter

Invalid sub-directory entry

Invalid working directory

Processing cannot continue

Probable non-DOS disk

Continue (Y/N)?

Unrecoverable error in directory

Convert directory to file (Y/N)?

Examples

```
CHKDSK /F 
```

checks the directory of the current disk, displays the errors, asks if you want to fix them, and acts accordingly.

```
CHKDSK B: /V 
```

checks the directory of the disk in Drive B, displaying the name of every directory and file on the disk, and reports on progress.

```
CHKDSK B:>\USER\TOM\errors 
```

checks the directory of the disk in Drive B and outputs any errors to the file Errors in the USER\TOM directory that is on the current disk.

CLS (Clear Screen)

Internal

CLS

Clears the screen.

Parameters

None

Notes and Suggestions

- Use the CLS command to clear a cluttered, hard-to-read screen.

Examples

CLS

clears the monitor screen, and moves the prompt and the cursor to the *home* position (the upper left corner of the screen).

COMMAND

External

COMMAND [*pathname*] [*device*] [/E:*size*] [/P]
[/C *string*]

Starts a new command processor (the MS-DOS program that contains all the internal commands).

The command processor is loaded into memory in two parts, the *transient* part and the *resident* part. Some application programs write over the transient part of COMMAND.COM when they run. When this happens, the resident part of the command processor looks for the COMMAND.COM file on disk so it can reload the transient part.

Parameters

pathname specifies the drive and directories in which the command processor is to look for the COMMAND.COM file if it needs to reload the transient portion of the file into memory.

device specifies a different device for input and output. It can be:

AUX to specify an auxiliary device, usually RS232 Serial Port 1.

COM1 to specify RS232 Serial Port 1.

COM2 to specify RS232 Serial Port 2.

CON to specify the console (keyboard input, screen output).

E:*size* specifies the environment size, in bytes. *size* is in the range 128 to 32768. The default is 128.

If *size* is fewer than 128 bytes, MS-DOS uses 128 bytes. If *size* is more than 32768 bytes, MS-DOS uses 32768 bytes. In either case, MS-DOS displays:

Invalid environment size specified

/P tells the command processor not to exit to a higher level.

/C string tells the command processor first to execute the command or commands specified by *string*, then to return. The */C* switch is valid only as the last parameter.

Example

```
COMMAND /C CHKDSK B: 
```

tells the command processor to start a new command processor under the current program, run the command CHKDSK B:, and then return to the first command processor.

Refer to the sample CONFIG.SYS file in Appendix C for an example of how to use a pathname and the */P* switch with COMMAND.

COPY

Internal

COPY *source pathname* [*target pathname*]
[/A] [/B] [/V]

Copies the contents of one disk file (*source pathname*) to another disk file (*target pathname*), either creating or overwriting the target. The target disk must be formatted, and the target directory must exist. The source file remains intact; it is copied, not moved.

By varying the syntax of COPY slightly, you can also use COPY to:

- Add a file or files to the end of an existing file.
- Combine existing files into a new file.

For more information, see the sections on appending and combining, following the examples of the regular COPY command.

Parameters

*source
pathname* specifies the file you want to copy.

*target
pathname* specifies the name and destination you want to give the copy.

If *target pathname* already exists, COPY overwrites the existing file. If *target pathname* does not exist, COPY creates it.

If you omit the filename part of the *target pathname*, COPY assumes you want to give the copy the same name as the source. You can do this as long as you are copying the file to a different directory. For example, you can use the command `COPY A:\SALES\firstqtr B:\SALES`. You cannot do this, however, if you are copying to the same directory, because MS-DOS does not let you “copy a file to itself.” For example, you cannot use either the command `COPY B:memos B:memos` or the command `COPY memos`.

/A When used with the source file, /A tells MS-DOS to treat the file as an ASCII file (also called a *text* or *data* file). Therefore, MS-DOS copies only to the first end-of-file character. (If you create a file with EDLIN, this character is Control-Z.)

When used with the target file, /A tells MS-DOS to add an EOF character to the end of the file.

/B When used with the source file, /B tells MS-DOS to treat the file as a binary file, such as a program file. Therefore, MS-DOS copies the entire file.

When used with the target file, /B tells MS-DOS not to add an EOF character to the end of the file.

Each of the /A and /B switches affects the file immediately preceding it on the command line and all files following, until the file preceding the next switch.

For example, if you type this:

```
COPY thisfile /A thatfile 
```

the /A parameter affects both files. If you type this:

```
COPY thisfile /A thatfile /B 
```

the /A parameter affects only the file Thisfile.

If you omit the /A and /B switches, MS-DOS uses /B.

/V tells MS-DOS to verify that the sectors written to the disk are recorded properly. This parameter slows the process because MS-DOS must check each entry recorded on the disk.

Notes and Suggestions

- Use the /B parameter with a target file to remove the EOF character from the file. Some application programs require that the EOF character not be included.
- Use COPY to:
 - Copy a file to another disk. COPY lets you make copies of files to protect important information.
 - Reduce *file fragmentation*. Fragmentation occurs as a result of extensive file creating and deleting, and causes wasted disk space. Copying your files to another disk is the only way to reduce file fragmentation.
 - Transfer external commands to another disk. When you receive your system disk, all external commands are in the ROOT directory. If you want, you can tailor your directory structure to your own needs by moving some of the commands to their own directory.

Examples

```
COPY *.* /A B: [ENTER]
```

copies all the files from A:\ to B:\

```
MKDIR \BIN [ENTER]  
COPY format.com \BIN [ENTER]
```

transfers the Format.com external command to another directory by creating the directory \BIN in the ROOT directory of the system disk in Drive A and copying Format.com into it. You can now copy any other .com files you use regularly into the \BIN directory.

```
COPY \personal.dat B: [ENTER]
```

copies the file Personal.dat from the ROOT directory of Drive A to the home directory of Drive B.

```
COPY memos.txt /A B:corr.txt 
```

copies the file `Memos.txt` from the current directory to the home directory of the disk in Drive B, naming the new file `Corr.txt`. MS-DOS copies information only up to the first EOF character, and adds an EOF character to the end of the new file.

```
COPY B:taxes85.dat /A taxes84.dat 
```

copies the file `Taxes85.dat` from the home directory in Drive B to the current directory, naming the new file `Taxes84.dat`. MS-DOS copies information only up to the first EOF character, and adds an EOF character to the end of the new file.

```
COPY prog.exe B:prog1.exe 
```

copies the file `Prog.exe` that is in the current directory to the home directory of Drive B. MS-DOS does not add an EOF character to the end of the new file, `Prog1.exe`.

```
COPY *.1st/A combin.prn 
```

finds all files that have the extension `.1st` in the current directory, and copies them to the new file `Combin.prn` in the current directory. MS-DOS copies information only up to the first EOF character, and adds an EOF character to the end of the new file.

Using COPY to Combine Files

COPY *source pathname1* [+ *source pathname2* . . .]
[*target pathname*] [/A] [/B] [/V]

The above variation of the COPY syntax combines any number of source files into a target file, which the COPY command creates. The target file must have a unique *pathname*. Otherwise, COPY overwrites the existing file.

Parameters

/A, /B, and /V These parameters have the same functions as they do for the regular COPY command. **The only difference is that if you omit /A and /B when combining files, COPY uses /A.**

Notes and Suggestions

- **Warning:** Never combine files into a target file that has the same name as an existing file. Instead, append other files to the existing file.

Example

```
COPY oldlist1.dat + oldlist2.dat newlist.dat  
ENTER
```

The above command combines the files Oldlist1.dat and Oldlist2.dat, which are both in the current directory:

Oldlist1.dat	Oldlist2.dat
Jessica Short	Anne Barnes
Mike McAdam	Larry Thomas

It places the contents of both in a new file, Newlist.dat, also in the current directory:

Newlist.dat

Jessica Short

Mike McAdam

Anne Barnes

Larry Thomas

If you wish, you can now sort Newlist alphabetically, using the SORT command.

Using COPY to Append Files

COPY *target pathname* + *source pathname1*
[+ *source pathname2* . . .] [/A] [/B] [/V]

The above variation of the COPY syntax appends any number of source files to the end of an existing target file, in the order in which you list the files.

Parameters

/A, /B, and /V These parameters have the same functions as they do for the regular COPY command. The only difference is that if you omit /A and /B when appending files, COPY uses /A.

Example

```
COPY year.dat + qtr2.dat ENTER
```

The above command appends the file Qtr2.dat to the file Year.dat:

Year.dat (before)	Qtr2.dat
Jan 130 units	Apr 140 units
Feb 145 units	May 150 units
Mar 110 units	Jun 165 units

Year.dat (after)

Jan 130 units
Feb 145 units
Mar 110 units
Apr 140 units
May 150 units
Jun 165 units

CTTY (Change I/O Device)

Internal

CTTY *device*

Lets you change the device from which you issue commands. (The letters TTY represent the keyboard, the device you normally use.)

Parameters

device specifies the new I/O device. It can be:

AUX (auxiliary), which refers to the first RS232 serial port.

COM1 or **COM2** (communications), which refer to the first and second RS232 ports, respectively.

CON (console), which refers to the keyboard for input and to the screen for output. Use this option to switch back to standard I/O.

device cannot be PRN (printer), because the printer is not capable of input.

Notes and Suggestions

- Many programs do not use MS-DOS for input, output, or both. They “talk” directly to the hardware. CTTY does not affect these programs.

Examples

```
CTTY AUX 
```

moves all command input/output (I/O) from the current device to the auxiliary device.

```
CTTY CON 
```

changes the input device to the keyboard and the output device to the screen.

DATE

Internal

DATE [*mm/dd/yyyy*]

Enters or changes the system date. MS-DOS records this date in the directory for any files you create or change. You can also use the DATE command to display the current date.

MS-DOS is programmed to change the months and years correctly, taking into account leap years and the number of days in the months.

Parameters

mm/dd/yyyy specifies the date in numerical form.

mm (month) is a 1- or 2-digit number in the range 1-12.

dd (day of month) is a 1- or 2-digit number from 1-31.

yyyy (year) is a 2-digit number in the range 80-99 (with 1980 assumed) or a 4-digit number in the range 1980-2099.

If you omit the *mm/dd/yyyy* parameter, DATE displays the current date, and asks you to enter the new date. If you do not want to change the date, press **ENTER**. If you do want to change it, enter it in the *mm/dd/yyyy* format.

Notes and Suggestions

- If the month or day is a number less than 10, you do not need to include a leading zero. (For example, you can enter 9/9/84.) When MS-DOS stores and displays the date, it includes a leading zero in a 1-digit day and excludes it from a 1-digit month. (For example, it stores 9/09/1984.)
- You can separate the date, month, and year with either slashes or hyphens.

- You can change the date from the keyboard or from a batch file. (Normally, MS-DOS displays a date prompt each time you start up your system. It does not, however, if you use an Autoexec.bat file. Therefore, you might want to include a DATE command in that file.)
- Using the COUNTRY command in the CONFIG.SYS file, you can change the date format to the European standard *dd-mm-yy*. Refer to Appendix C, "Configuring Your System," for more information on the CONFIG.SYS file.
- When you change the date known to the system, you also change the date in any application program you use. This can be very handy.

Suppose you have a program that keeps track of purchase orders according to the date received. For some reason, you get behind and cannot enter the information on that date. Simply enter it later, after *turning back the calendar* to the necessary date.
- You can also use DATE and TIME to include the date and time on a printout. For example, press **CTRL** **P** to send all output to the line printer, as well as to the screen. Then type:

```
DATE ENTER ENTER
TIME ENTER ENTER
```

Press **CTRL** **P** again to turn off the print function. Now use **SHIFT** **PRTSC**, **CTRL** **P**, or the PRINT command as necessary to make your printout.

Examples

```
DATE ENTER
```

displays the current date, and prompts you for the new date. For example, if the date is July 16, 1985, MS-DOS displays:

```
Current date is Mon 7-16-1985
Enter new date:
```

You can now change the date or press to bypass the prompt.

DATE 07/14/1985

enters the current date as Saturday, July 14, 1985.

DEL (Delete)

Internal

See the ERASE command.

DIR (Directory)

Internal

DIR [*pathname*][*/P*][*/W*]

Displays information about: (1) files in the current directory, or (2) files in the directory specified by *pathname*, or (3) the one file specified by *pathname*. Files are listed with their size (in bytes) and the date and time they were last modified.

Parameters

pathname can specify an entire drive, a directory, or a file.

The standard MS-DOS defaults apply. If you omit the drive, MS-DOS assumes the current drive. If you omit the directory, it assumes the current directory.

You can use MS-DOS wildcards in the *pathname*, and can shorten the commands even more by using commands that DIR accepts as equivalent to the wildcard commands:

DIR	=	DIR *.*
DIR <i>filename</i>	=	DIR <i>filename</i> .*
DIR <i>.ext</i>	=	DIR *.* <i>.ext</i>

/P selects the *page* mode. With */P*, display of the directory pauses when the screen is filled. It displays this message:

Strike a key when ready...

To resume display of output, press the space bar.

/W selects a wide display. With */W*, MS-DOS displays filenames only; it does not display sizes or modification dates.

Notes and Suggestions

- DIR first causes MS-DOS to display the disk's volume label, which was assigned during formatting. If you did not assign a label, MS-DOS displays a message to that effect. Then MS-DOS lists each file with its size (in bytes) and the time and date of the last modification. It then gives the number of files listed and the number of bytes remaining on the disk.
- Note that if the COUNTRY command in the CONFIG.SYS file is set to a country other than the U.S., the directory date and time formats differ. Refer to Appendix C, "How to Configure Your System," for more information on the CONFIG.SYS file.
- DIR cannot list the directory if you specify a pathname of more than 64 characters (including the drive name). Therefore you might need to change directories if you want to list files in deep subdirectories.

Examples

```
DIR B:\USER\mailsr.t.bat 
```

provides information on how much space B:\USER\Mailsrt.bat takes on the disk in Drive B and how much space remains.

```
DIR B:\ 
```

tells you the name of the files in Drive B. The backslash indicates the ROOT directory. If you do not use it, MS-DOS lists the files in the home directory. If the file you want is not in the ROOT directory, use the DIR command as needed to check the subdirectories.

```
  DIR 
```

sends a listing of the directory to the printer, as well as to the screen, until you again press .

DIR B:

displays a list of all files in the home directory of Drive B.

DIR /W

displays, in the wide mode, the names of all files in the current directory.

DIR \USER*.bat/P

displays a list of all batch files in the \USER directory on the current drive. MS-DOS halts (pauses) the display when the screen is full. Press the space bar to continue.

DIR A:\USER\acct.*

displays a list of files that have the name Acct—regardless of their extensions—and that are in the A:\USER directory.

DISKCOMP

External

DISKCOMP [*drive1*] [*drive2*] [/1] [/8]

Compares the contents of two diskettes.

Parameters

- | | |
|---------------|--|
| <i>drive1</i> | is the drive containing the source diskette. |
| <i>drive2</i> | is the drive containing the target diskette. If you omit either drive specification, DISKCOMP uses the current drive. If the source and target drives are the same, DISKCOMP does a single-drive comparison, prompting you to swap diskettes as necessary. |
| /1 | causes DISKCOMP to compare only the first side of two double-sided diskettes. If you omit /1, DISKCOMP compares both sides. (If both diskettes are single-sided, you do not need to specify /1.) |
| /8 | causes DISKCOMP to compare only the first 8 sectors of each track. If you omit /8, DISKCOMP automatically compares either 9 or 15 sectors, according to the format of the two diskettes. |

Notes and Suggestions

- The main purpose of this command is to compare diskettes after a DISKCOPY operation to ensure that the diskettes contain identical data.
- If you are comparing a diskette with a backup diskette created using COPY or XCOPY, the diskettes usually are not identical. This is because COPY and XCOPY duplicate the information, but don't necessarily place it in the same location on the target diskette. Use the FC utility, instead of DISKCOMP, to compare such diskettes.
- DISKCOMP does not work with hard disks.

- You cannot use DISKCOMP over a network. You also cannot use it with drives that are assigned, joined, or substituted.
- DISKCOMP performs a track-by-track comparison. It automatically determines the number of sides and sectors-per-track based on the format of the source disk. Therefore, the target diskette must have the same number of sides and sectors-per-track as the source. If the target is **not** the same type as the source, DISKCOMP displays the message:

```
Drive types (double, single-
sided) or diskette types not
compatible
```

If all the tracks are the same, DISKCOMP displays:

```
Diskettes compare OK
```

If the tracks are not the same, DISKCOMP displays a Compare error message that includes the track and side number where it found the mismatch.

- When the comparison is complete, DISKCOMP prompts:

```
Compare more diskettes (Y/N)?
```

If you press **[Y]** for another DISKCOMP operation, the comparison is done on the drives you originally specified, but it is now based on the sector and side configuration of the new diskettes. For example, if the new diskettes are double-sided, and the previous diskettes were single-sided, DISKCOMP does a double-sided comparison. If you press **[N]**, the DISKCOMP function terminates.

- If you end DISKCOMP, and if the disk in the current drive does not contain MS-DOS, DISKCOMP prompts you:

```
Insert disk with COMMAND.COM in
drive x and strike any key when
ready
```

- You can escape from the DISKCOMP command by pressing **CTRL** **C**. The command delays acting on the **CTRL** **C** request until it is not performing a diskette read.
- If a read error occurs on a track of one diskette, that track is reported as being different.

Examples

DISKCOMP **ENTER**

compares two diskettes for identical data on the current drive. You are prompted when to swap diskettes.

DISKCOMP A: B: /1 /8 **ENTER**

compares the diskette on Drive A with the diskette on Drive B. DISKCOMP compares one side of the diskettes and eight sectors of each track.

DISKCOPY

External

DISKCOPY [*source drive*] [*target drive*]

Makes an exact copy of the diskette in the *source drive* on the diskette in the *target drive*. DISKCOPY automatically formats the diskette in the *target drive* if it is not already formatted exactly like the *source* diskette.

Parameters

source drive is the drive containing the disk you want to copy.

target drive is the drive containing the disk to receive the copy.

If you omit either drive specification, MS-DOS uses the current drive.

Therefore, if you omit both drive specifications, MS-DOS does a single-drive copy on the current drive. (This assumes that the DISKCOPY.COM file is in the current directory, so that MS-DOS can find it, or that you use PATH beforehand to tell MS-DOS where to look for the file.)

Notes and Suggestions

- DISKCOPY only works on floppy diskettes.
- If the source and target drives are the same, MS-DOS performs a single-drive copy. It prompts you to insert the disks at the appropriate times. Press the space bar to continue.
- After copying, DISKCOPY prompts:

```
Copy complete
Copy another (Y/N)? __
```

If you press **[Y]**, MS-DOS prompts you to insert the disks. Then it performs the next copy, using the drives you specified earlier.

If you press **[N]**, MS-DOS exits the DISKCOPY command.

- By using DISKCOPY to duplicate your MS-DOS diskette, you can create another operating system diskette. It is a good idea to make copies of all your diskettes—especially system and application program diskettes—then store them in a safe place.
- The source and target diskettes must have the same number of tracks and the same number of sectors per track. (See the FORMAT command for this information.)
- DISKCOPY does not aid in making fragmented disk files contiguous. Use COPY or XCOPY for this purpose.
- You cannot use DISKCOPY over a network.

Examples

DISKCOPY

copies all information from a diskette in the current drive to another diskette in the current drive. MS-DOS prompts you to insert diskettes as necessary.

DISKCOPY A: B:

copies all information from the diskette in Drive A to the diskette in Drive B. Before starting DISKCOPY, MS-DOS prompts you to insert diskettes.

DISKTYPE

External

DISKTYPE [*drive*]

Displays information on the size and capacity of the indicated drive.

Parameters

drive is the disk drive for which you wish to determine the type. It can be either a floppy drive or a hard drive. If you omit *drive* DISKTYPE displays information about the current disk.

Notes and Suggestions

- Using DISKTYPE with floppy diskettes displays the number of sides, tracks, and sectors-per-track of the diskette, and the FORMAT command used to format the diskette.
- Using DISKTYPE with a hard disk displays the number of heads, cylinders, and sectors-per-track.

Examples

```
DISKTYPE A: 
```

displays a message similar to the following:

```
Diskette is formatted with 2 sides, 40 tracks, 9
sectors/track
---> FORMAT A:/4
---> FORMAT B:
```

```
DISKTYPE C: 
```

displays a message similar to the following:

```
Fixed disk contains 4 heads, 200 cylinders, 17
sectors/track
```

ECHO

Internal

ECHO [ON|OFF]*message*

Turns the batch ECHO feature on or off, and lets you use messages while the batch files are executing.

Parameters

ON	tells MS-DOS to display batch file commands.
OFF	tells MS-DOS not to display batch file commands.
<i>message</i>	is text for MS-DOS to display whenever it encounters an ECHO command. If you do not specify ON or OFF, and you omit a message, MS-DOS displays the current setting of ECHO.

Notes and Suggestions

- Normally, the screen displays (echoes) commands in a batch file as you run the file. ECHO OFF turns off this feature. ECHO ON turns it on again. (**Note:** ECHO is automatically on again after the batch file executes.)
- ECHO *message* displays the specified message, regardless of whether ECHO is on or off.
- Specifying ECHO OFF at the MS-DOS prompt turns off the system prompt.
- Whenever you want to protect information in a batch file, use ECHO OFF. You can also use ECHO OFF when you do not want to clutter the screen with unnecessary information.
- Turn ECHO off, and insert your own messages to be displayed while a batch file is executing.
- The following batch file formats a disk in Drive B, and then checks the disk. (The file uses some external commands; so, it must be created in a directory that contains the external commands.)

```
COPY con prepdisk.bat 
ECHO OFF 
REM This formats and checks disks
in Drive B 
REM It is called prepdisk.bat

FORMAT B: 
ECHO Do you want to check Drive
B? 
PAUSE 
CHKDSK B: 
ECHO ON 
 
```

When you run the file, MS-DOS displays:

```
ECHO OFF
```

The **FORMAT** copyright displays and, as the batch file runs, the screen shows:

```
Insert new diskette for drive B:
and strike any key when ready...

Formatting...Format Complete
  nnnnnn bytes total disk space
  nnnnnn bytes available on disk

Format another (Y/N)? Do you want
to check Drive B?

Strike a key when ready...

  nnnnnn bytes total disk space
  nnnnnn bytes available on disk

  nnnnnn bytes total memory
  nnnnnn bytes free
```

As you can see, MS-DOS does not echo the commands. The screen does display the prompt Do you want to check Drive B? This is because the **ECHO** command used here includes the message parameter.

ERASE

Internal

ERASE *pathname*

DEL *pathname*

Erases (deletes) a file or files from the current directory or from the directory specified by *pathname*. By erasing unnecessary files, you can create more free space in a directory.

Parameters

pathname specifies the file you want to delete.

Notes and Suggestions

- **Warning:** If you omit the filename from the *pathname*, MS-DOS assumes you want to erase all files in the directory. (Using the wild card *.* as the filename is the same as omitting the filename.)
- If you use the full wild card (*.*), ERASE displays the prompt: Are you sure (Y/N)? If you type Y , MS-DOS erases the files. If you type N , MS-DOS exits the command.
- If you omit the entire *pathname*, ERASE returns an error message.
- To avoid erasing important files, use the DIR command with the appropriate wild card to check which files you need.
- Use ERASE to:
 - Erase or delete files in an old directory after copying them to a new directory.
 - Delete all the files in a directory so that you can delete the directory.
 - Erase a file for which you have no further need.

Examples

```
ERASE *.1st 
```

can be used after you combine the files A.1st and B.1st from the current directory into the new file All.1st on a different directory. It erases all files with the extension .1st in the current directory. Be sure that there are no other files you want to keep that have the .1st extension.

Warning: If you include an extension, such as the .1st used here, MS-DOS does not prompt you before erasing the files, regardless of whether use a wild card. It prompts only when you use the full wild card (*.*) .

```
ERASE \BIN\USER\MARY\text.txt 
```

erases the file Text.txt from the directory BIN\USER\MARY on the current disk.

```
ERASE B:\USER\JOHN\*.* 
```

tells MS-DOS to delete all the files in the directory \USER\JOHN on Drive B. MS-DOS asks: Are you sure? (Y/N)?

```
ERASE B:\BIN\USER\MARY 
```

tells MS-DOS to erase all files in the \BIN\USER\MARY directory on Drive B. MS-DOS prompts: Are you sure (Y/N)?

```
ERASE file2 
```

erases the file File2 from the current directory.

EXE2BIN (Executable to Binary) External

EXE2BIN *source pathname* [*target pathname*]

Converts an executable (.exe) file to a binary (.bin) file format. EXE2BIN is an advanced command, recommended for experienced users only. The source file must be in valid .exe format produced by the linker. The resident, or actual code and data part of the file, must be less than 64K. There must be no STACK segment.

Parameters

<i>source pathname</i>	specifies the executable file to be converted. If you do not include an extension in the source pathname, the extension defaults to .exe.
<i>target pathname</i>	specifies a new, binary-format file to receive the converted program file. If you omit the drive, MS-DOS uses the drive specified in the source pathname. If you omit the extension, it gives the new file the extension .bin. If you omit the entire target pathname, MS-DOS uses the source pathname.

Notes and Suggestions

- Converting executable files to binary format saves space and speeds program loading. Two kinds of conversions are possible, depending on whether the initial CS:IP (Code Segment:Instruction Pointer) is specified in the .exe file.
- If CS:IP is not specified, EXE2BIN assumes a pure binary conversion. If segment fixups are necessary (the program contains instructions requiring segment relocation), EXE2BIN requests a fixup value. This value is the absolute segment at which the program is to be loaded.

In a pure binary conversion, the resulting program is usable only when loaded at the absolute memory address specified by a user application. The command processor cannot properly load the program.

- If CS:IP is specified as 0000:100H, EXE2BIN assumes the file is to be run as a .com file with the location pointer set at 100H by the assembler statement ORG. It deletes the first 100H bytes of the file. It allows no segment fixups, as .com files must be segment relocatable.

After this kind of conversion is complete, you can rename the resulting file, giving it a .com extension. Then, the command processor can load and execute the program in the same way as it does the .com programs supplied on your MS-DOS disk.

- If CS:IP does not meet either criterion (discussed in the preceding “Notes and Suggestions”)—or if it meets the .com file criterion but has segment fixups—EXE2BIN displays an error message. EXE2BIN also displays an error message if the file is not a valid executable file.

Examples

```
EXE2BIN testfile.exe B: 
```

converts the file Testfile.exe that is in the current directory to binary format, and places the new file, Testfile.bin, in the home directory of Drive B.

```
EXE2BIN A:\USER\oldfile.exe A:newfile 
```

converts the file Oldfile.exe that is in A:\USER to binary format, and places the new file, Newfile.bin, in the home directory of Drive A.

EXIT

Internal

EXIT

Returns control from the MS-DOS command level to a previous level, if one exists. If you have entered the MS-DOS command level from an application program, use EXIT to return to the application.

Parameters

None

Notes and Suggestions

- While running an application program, you might need to use an MS-DOS command. To do so, you must first call the command level from your program. Then, enter the command you wish to use. When the command finishes executing, enter EXIT to return to your application program.

Examples

Call the command level from your program, type your command, and then use EXIT to return.

```
DIR B:   
EXIT 
```

lists the directory of Drive B; then returns to your application program.

FC (File Comparison)

External

FC [/A] [/B] [/C] [/L] [/LB*number*] [/N] [/T] [/W]
[/*number*] *pathname1* *pathname2*

Compares the contents of two files or sets of files.

Parameters

pathname1 and *pathname2* specify the files to compare. By using a wildcard (such as File?.txt), you can compare sets of files, instead of individual files.

/A abbreviates the output of an ASCII comparison. Normally, FC displays only the matching lines that precede and follow each set of differences. It uses an ellipsis (...) to represent the intermediate (non-matching) lines. (See “Notes and Suggestions,” below, for more information on how FC reports differences.)

/B forces a binary comparison of the files. FC compares the two files byte-by-byte, and makes no attempt to resynchronize after a mismatch. It displays mismatches in the following format:

```
xxxxxxx yy zz
```

where *xxxxxxx* is the relative address of the pair of bytes (from the beginning of the file). Addresses start at 00000000. *yy* and *zz* are the mismatched bytes from *pathname1* and *pathname2*, respectively. If one file contains less data than the other, FC displays a message, telling you which is longer.

/B is the default when you compare .exe, .com, .sys, .obj, .lib, or .bin files.

/C causes the matching process to ignore the case of letters, interpreting them as all uppercase. For example:

Much MORE data IS NOT FOUND

matches:

much more data is not found.

Use **/C** in source file comparisons only.

/L compares the files in ASCII mode. This is the default when you compare files that do not have the .exe, .com, .sys, .obj, .lib, or .bin extension.

/L*number* sets the internal line buffer to the specified *number* of lines. Files that have more than this number of consecutive, non-matching lines cause the comparison to cancel. If you omit this parameter, FC uses 100.

/N displays the line numbers in an ASCII comparison.

/T does not expand tabs to spaces. The default is to treat tabs as spaces to eight column positions.

/W causes FC to compress *white spaces* (tabs and spaces) during the comparison. When you use **/W**, FC interprets multiple contiguous white spaces as one white space. Note that although FC compresses white spaces, it does not ignore them. (The two exceptions are any beginning and ending white spaces on a line, which FC **does** ignore.)

For example (an underscore represents a white space):

____More__data__to__be__found____

matches:

More_data_to_be_found

and:

____More____data__to__be____found

but does not match:

____Moredata__to__be__found

Use the /W parameter in source file comparisons only.

/number

specifies the number of lines that must match for the file to be considered as matching after FC finds a difference. *number* can be in the range 1 to 9. If you omit *number*, FC uses 2. Use this parameter in source file comparisons only.

Notes and Suggestions

- The two kinds of comparisons are line-by-line and byte-by-byte.
- In a line-by-line comparison, FC marks off blocks of non-matching lines, beginning each block with the last matching line, and ending it with the next matching line. Following is an example.

Suppose your current directory contains these source files, in which each letter represents a program line:

File1.src	File2.src
a	a
b	b
d	g
e	h
f	k
k	m
m	o
n	p
o	
p	

FC blocks the above files as follows:

File1.src	File2.src
a	a
b	b
d	g
e	h
f	k
k	
m	m
n	o
o	
p	p

For each block that contains a mismatch, FC displays:

```
***** filename1
the last line that matches in the two files
the intermediate, non-matching lines in
first file
the next line that matches in the two files
***** filename2
the last line that matches in the two files
the intermediate, non-matching lines in
second file
the next line that matches in the two files
*****
```

Therefore, for the above files, it displays the following. (The notes do not appear.)

```
***** File1.src
```

```
b
```

```
d
```

```
e
```

```
f
```

```
k
```

```
***** File2.src
```

```
b
```

```
g
```

```
h
```

```
k
```

```
*****
```

```
***** File1.src
```

```
m
```

```
n
```

```
o
```

```
***** File2.src
```

```
m
```

```
o
```

```
*****
```

Note: File1
contains bdefk
where File2
contains bghk.

Note: File1
contains mno
where File2
contains mo.

- In a byte-by-byte comparison, FC displays the bytes that are different. The following is the output from a comparison of the File1.src and File2.src files (above). It is produced with the command `FC /B file1.src file2.src`.

```
00000006: 64 67
```

```
00000009: 65 68
```

```
0000000C: 66 6B
```

```
0000000F: 6B 6D
```

```
00000012: 6D 6F
```

```
00000015: 6E 70
```

```
fc: File1.src longer than File2.src
```

- If FC produces too much output to appear on one screen at a time, use the MORE filter to display the output one screen at a time. For example, type:

```
FC /B File1.src File2.src | MORE
```

```
ENTER
```

- You can redirect FC's output to a file by appending *>target pathname* to the end of the FC command. For example, to send the output of the above command to the file Differ.src in the current directory, type:

```
FC /B File1.src File2.src >
Differ.src [ENTER]
```

- If the number of lines in the internal buffer is smaller than the number of consecutive, differing lines, FC stops. It displays:

```
resynch failed. Files are too
different
```

Then, it returns to the MS-DOS system prompt.

Examples

```
FC /B test1.src test2.src >test3.src [ENTER]
```

does a binary comparison of the files Test1.src and Test2.src that are in the current directory, and redirects output to the file Test3.src that is also in the current directory.

```
FC /4/W/C B:\USER\myfile.src
B:\USER\myfile1.src [ENTER]
```

does a line-by-line comparison of B:\USER\Myfile.src and B:\USER\Myfile1.src, compressing white spaces and ignoring case. After FC finds a mismatch, at least four lines in a row must match for the file to be considered matching.

FDISK

External

FDISK

Creates a maximum of four disk partitions within the first 32 megabytes of a hard disk. Also used to change or delete those partitions, and to display information about them.

FDISK is part of the procedure for initializing a hard disk:

1. Use the Format Hard Disk utility (on the utilities diskette) or the HSECT command to format the hard disk.
2. Use FDISK to create partitions on the disk.
3. Use FORMAT to format the partitions.

Normally, you need to do this procedure only once.

Parameters

None

Notes and Suggestions

- If your computer supports more than one operating system, you can partition your hard disk so that you can install more than one operating system on the disk. You might also want to partition it for other specialized applications. Using FDISK, you determine where and how much of the first 32 megabytes to allocate for each partition. (To use space above 32 megabytes, you need to use MLPART after using FDISK.)
- Even if you do not intend to use a system other than MS-DOS, or if you don't know which systems you might use in the future, you need to enter the FDISK command as part of the initialization procedure.

- When you enter the FDISK command, the screen displays the following menu:

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

Enter Selection -->

Press ESC to exit to MSDOS

You can create a maximum of four partitions on any hard disk. After you type 1 to select the Menu Option 1, the screen displays:

Do you wish to use the entire
hard disk for DOS (Y/N)-->Y

If you want to use all space (through the first 32 megabytes) for MS-DOS, or if you don't know which other systems you might use later, type Y when asked if you want to use the entire hard disk for DOS. FDISK creates the DOS (MS-DOS) partition, and automatically makes that partition *active* so that you can boot from it. It then displays:

System needs to reboot
Insert system disk in Drive A
Press any key to reset the system

Once you reboot the system, you are ready to format the DOS partition, using the FORMAT command. Use FORMAT /S if you want to boot from the hard disk. (See the FORMAT command for more information.)

- If you want multiple partitions, type N when asked if you want to use the entire hard disk for DOS. The screen displays the amount of space available, and asks for the size (in cylinders) and location (starting cylinder) of the first partition. After you provide the numbers, you need to make the DOS partition active if you want to boot from the hard disk. Use Menu Option 2 to do this.

After you make the DOS partition active, FDISK instructs you to reboot the system. Once you do so, you are ready to format the DOS partition, using the FORMAT command. Use FORMAT /S if you want to boot from the hard disk. (See the FORMAT command for more information.)

- You can create a DOS partition on Drive D, but it cannot be bootable.
- Partitions that you create using other operating systems appear as *non-DOS* on the FDISK status screens.
- If you use more than one operating system on your computer, you might want to change the active partition. This is the partition that contains the operating system and files you access whenever you turn on or reset the computer. To change the active partition, use Menu Option 2, and specify the number of the partition you want to make active. The screen displays the number and status (A for active, or N for non-active) of the various partitions.
- If you ever want to change the size of your DOS partition, you need to delete and recreate it. To delete the partition, use Menu Option 3.

This option warns you that it deletes all data in the partition you specify. Before proceeding, use the BACKUP command to back up your files to floppy diskettes. After repartitioning the disk, use RESTORE to replace the files onto the hard disk.

- Use Menu Option 4 to display information about your hard disk partitions. The screen displays the partition's number, status (A for active or N for non-active), type (DOS or non-DOS), starting cylinder, ending cylinder, and size (in cylinders).
- If you have already installed Drive C, choose the Menu Option 5 to verify the status of Drive D.
- Use **[ESC]** to return to FDISK's main menu from the other FDISK screens and to exit from the main menu.

Examples

FDISK **[ENTER]**

loads FDISK from the current drive and displays the FDISK menu. To partition Drive C, press **[1]**. The screen displays:

```
Do you wish to use the entire hard
disk for DOS (Y/N)--> Y
```

To select one partition, press **[Y]**. If you have a specialized need for selecting more than one partition, enter the partition size in number of cylinders and the beginning cylinder as prompted.

B: **[ENTER]**

FDISK **[ENTER]**

establishes Drive B as the current drive, loads FDISK from Drive B, and displays the FDISK menu. To verify the status of Drive C, select Menu Option 4. The screen display shows information such as:

Partition	Status	Type	Start	End	Size
1	N	DOS	0	49	50

The status column refers to active or non-active. An active partition boots directly from Drive C when the system is started. The other columns indicate that there is one partition, beginning at Cylinder 0 and ending at Cylinder 49, for a total size of 50 cylinders, and that the partition contains a disk operating system.

FIND

External

FIND [/V] [/C] [/N] "string" [pathname ...]

FIND is a filter that searches for a specified string of text in a file or files, specified by *pathname(s)*. Capitalization and punctuation in the string must be the same as in the file. Otherwise, MS-DOS cannot find the string.

Parameters

pathname is the file to search. If you omit *pathname*, MS-DOS searches for the string among the lines on the current screen display.

string is the text for which to search. You must enclose it in quotation marks. For example, to find all occurrences of the following string:

The whiteness of the whale

that are in the files Report1.txt and Report2.txt, type:

```
FIND "The whiteness of the whale"  
report1.txt report2.txt ENTER
```

MS-DOS displays the lines in the order in which you specify the files. If the string contains quotations, enclose each quotation in double quotation marks. For example, to find all occurrences of the following string:

The whiteness of the whale is a
recurring symbol in "Moby Dick."

that are in Report1.txt, type:

```
FIND "The whiteness of the whale  
is a recurring symbol in ""Moby  
Dick."" report1.txt ENTER
```

/V causes FIND to display all lines that do not contain the string. Do not use /V with the /C parameter.

- /C** causes FIND to display only the number of lines (in each file) that contain the string.
- /N** causes each line to be preceded by its relative line number in the file. Do not use /N with the /C parameter.

Notes and Suggestions

- Using FIND with the /N parameter, you can determine which lines contain the specified string. Then, knowing the line numbers, you can easily enter the EDLIN Edit Line command and change every occurrence of the string. (See Part 3, "Editing.")
- Suppose one of your suppliers, Johnson Auto Parts, buys out another, Samuel Parts. You need to change all occurrences of Samuel Parts to Johnson Auto Parts in your B:\USER\Inventory.dat file. To find the line number of each occurrence, type:

```
FIND /N "Samuel Parts"  
B:\USER\inventory.dat 
```

MS-DOS might display a list like this:

```
-----B:\user\inventory.dat  
  
[1]spark plugs          200 6-16-83 Samuel Parts  
[5]distributor caps    50  6-16-83 Samuel Parts  
[9]14" rad hose        25  6-16-83 Samuel Parts  
(2" dia.)
```

Now you can use EDLIN to edit Lines 1, 5, and 9 as needed.

Examples

```
FIND /C "- " B:\USER\profits.dat
```

displays the number of times a negative sign occurs in B:\USER\Profits.dat. In this way, you can see at a glance how many negative numbers you have in the file. (Notice the space after the negative sign. This is included so MS-DOS does not find hyphenated words.)

```
DIR B:| FIND /V "DAT"
```

displays all filenames in the home directory of the disk in Drive B that do not contain the string DAT.

FMAT2000

External

FMAT2000 *drive* [/S] [/V]

Formats a standard (5-¼-inch, double-sided) diskette for 720K bytes of data storage in a high-capacity floppy disk drive. We recommend using an 80-track, double-sided diskette (Cat. No. 26-410).

With Tandy 3000 series computers, you can use a 720K-format-diskette only in high-capacity drives.

Parameters

- drive* is the **high-capacity** drive that contains the diskette you want to format. Depending on your computer, *drive* can be either Drive A or Drive B or both. If you omit the drive specification, MS-DOS uses the current drive.
- /S causes FMAT2000 to copy the MS-DOS 3.2 system files to the diskette after formatting it. This makes the newly formatted diskette a system disk for a Tandy 3000 series computer.
- /V causes FMAT2000 to prompt for a volume label after formatting the disk. If you use this parameter, you can enter a label of a maximum of 11 characters, or you can press to bypass the prompt. Entering a volume label helps you keep track of your diskettes.

Notes and Suggestions

- If you have a high-capacity drive, use FMAT2000 (instead of FORMAT) to double the amount of data you can store on your standard diskettes.
- You can use FMAT2000 to format either a new diskette for data storage, or to **clear** an old diskette for data storage. Formatting any disk erases all information on the disk. So, be sure any old diskettes that you want to format do not contain information you need.

- MS-DOS prompts you for the diskette you want to format. For example, if you specify Drive A, MS-DOS asks you to:

Insert a new diskette for drive A
and strike ENTER when ready.

Remove the diskette currently in Drive A, and insert the diskette to be formatted. Press .
- Diskettes formatted with FMAT2000 are compatible with diskettes used with the Tandy 2000 computer. You can read and write to such diskettes, using a Tandy 2000. You cannot, however, use them to boot the Tandy 2000.

Example

```
FMAT2000 B: /S /V 
```

formats a standard diskette in a high-capacity Drive B for 720K bytes of storage, and prompts you for a label to assign the diskette.

FOR

Internal

FOR %c IN (set) DO command

(regular MS-DOS command)

FOR %%c IN (set) DO command

(batch file command)

Executes the specified *command* for each item in *set*.

Parameters

<i>c</i>	is any 1-character variable except 0-9.
<i>set</i>	can be a list of items, separated by spaces, or it can be one wild card item.
<i>command</i>	is the command to be executed. If you include %c or %%c at the end of <i>command</i> , MS-DOS sequentially substitutes each member of <i>set</i> in the command. Otherwise, it executes the command the appropriate number of times, but without substituting the members of <i>set</i> .

Notes and Suggestions

- Use FOR whenever you want to execute a command for several items so you do not repeat the command unnecessarily.

Examples

```
FOR %f IN (A:\ B:\ B:\USER) DO DIR %f [ENTER]
```

MS-DOS displays the listing for each of the three directories A:\, B:\, and B:\USER, in order.

```
COPY con 3dir.bat [ENTER]
REM This file displays directory listings for
%1, %2, and %3 [ENTER]
REM it is called 3dir.bat [ENTER]
FOR %%f IN (%1 %2 %3) DO DIR %%f [ENTER]
[F6] [ENTER]
```

substitutes other directories for those in the previous command by creating a batch file using replaceable parameters. (See "Creating a Batch File with Replaceable Parameters" in Part 1.)

To execute the preceding batch file, and see the listings for A:\BIN, A:\GAMES, and A:\ACCTS, type:

```
3dir A:\BIN A:\GAMES A:\ACCTS 
```

MS-DOS first substitutes the directory names, in order, for the replaceable parameters %1, %2, and %3. Then, it does a DIR command for each directory.

```
FOR %f IN (*.asm) DO TYPE %f 
```

displays all files in the current directory that have the extension .asm.

```
FOR %f IN (taxfile autofile homefile) DO DEL %f  

```

deletes the three files, in order, from the current directory.

FORMAT

External

FORMAT [*drive*] [/4] [/8] [/B] [/1] [/V] [/S]

Prepares the specified floppy diskette or hard disk to accept MS-DOS files. You can format either a blank disk or a disk that is already formatted. If the disk is already formatted, and you reformat it, you lose whatever programs and data are on the disk.

Formatting Floppy Diskettes

In formatting a floppy diskette, MS-DOS defines the diskette's *tracks* and *sectors*, and writes system information to the diskette.

When formatting certain types of diskettes in certain types of drives, the only parameter you **need** to specify is the drive. (You can specify other paramters, if you wish.) This applies to:

- Formatting a 5¼-inch, double-sided diskette in a 5¼-inch, double-sided drive.
- Formatting a 5¼-inch, high-density diskette in a 5¼-inch, high-capacity drive.
- Formatting a 3½-inch diskette in a 3½-inch drive. (In this case, you must have previously used the **DRIVPARM** command in your **CONFIG.SYS** file to inform the system of the existence of the 3½-inch drive. See Appendix C.)

In each of the above cases, **FORMAT** automatically *knows* how to format the diskette, based on the type of drive. For the other allowable drive/diskette combination (a 5¼-inch, double-sided diskette in a high-capacity drive), you must use either the /4 switch or the /8 switch. (See the "Parameters" section, below.)

After you enter the **FORMAT** command, the screen displays:

```
Insert new diskette for drive x:  
and strike ENTER when ready
```

Insert the diskette to be formatted, and press **ENTER**. **FORMAT** displays the numbers of the different disk areas as it formats them.

Formatting Hard Disks

In formatting a hard disk, **FORMAT** prepares a hard disk *partition* for use by MS-DOS. Before you can format a hard disk, you must do the following:

1. Use either the **HSECT** command (on your Supplemental Programs diskette) or the Format Hard Disk utility (on your utilities diskette) to define the disk's tracks and sectors.
2. Then, use the **FDISK** command (on your Supplemental Programs diskette) to partition the disk.

When formatting a hard disk, the only parameter you **need** to specify is the drive. (You can specify other parameters, if you wish.)

Before formatting the hard disk, **FORMAT** displays:

```
WARNING, ALL DATA ON NON-REMOVABLE DISK  
DRIVE x: WILL BE LOST!  
Proceed with Format (Y/N)?
```

If you want to format the disk, type **Y** . If you do not want to format it, type **N** .

Parameters

- drive* is either the hard disk drive you want to format or the floppy disk drive that contains a diskette you want to format. You must specify the drive.
- /4* causes a 5¼-inch, double-sided diskette in a high-capacity drive to be formatted as double-sided with 9 sectors per track and 360K bytes of storage.
- /8* causes a 5¼-inch, double-sided diskette (in either a standard, double-sided drive or a high-capacity drive) to be formatted with 8 sectors per track and 360K bytes of storage. This format is provided for compatibility with computers that format with 8 sectors per track.

- /B** does the same as /8. In addition, /B allocates space for you to later transfer the hidden system files to the diskette, using the SYS command. With /B, you can use SYS to place any version of MS-DOS on the disk. Without /B, you can use SYS only to place Version 3.2 on the disk. You cannot use /S or /V with the /B switch.
- /1** causes a 5¼-inch, double-sided diskette (in either a standard, double-sided drive or a high-capacity drive) to be formatted as single-sided with 180K bytes of storage. This format is provided for compatibility with computers that use single-sided formats.
- /V** causes FORMAT to prompt you for a volume label for the disk. By assigning a volume label, you protect the disk from accidental reformatting (erasure). Once you specify a label, FORMAT requires you to specify that label anytime you try to reformat the disk. A volume label can have a maximum of 11 characters. An example is PROGRAMS.
- /S** transfers the hidden operating system files to the disk, making the disk bootable.
- If the operating system isn't on the current drive, FORMAT prompts you to insert a system disk in the current drive (or in Drive A if the current drive is a hard disk).

Notes and Suggestions

- Formatting destroys any data on the disk.
- FORMAT ignores drive assignments created using the ASSIGN command.
- Do not use FORMAT with drives used with the ASSIGN, JOIN, or SUBST commands.
- You cannot format drives over a network.
- To ensure read/write reliability, use diskettes only in the type of drive in which you formatted them.

- The switches that you can use with the **FORMAT** command depend on the type of drive and (for floppy diskettes) the type of diskette. They are as follows:

Drive Type	Diskette Type	Valid Switches
5¼-inch double-sided	5¼-inch double-sided	/1 /8 /B /V /S
5¼-inch high-capacity	5¼-inch double-sided	/4 /1 /8 /B /V /S
5¼-inch high-capacity	5¼-inch high-density	/V /S
3½-inch	3½-inch	/V
hard disk		/V /S

- Once you format a hard disk, **FORMAT** prompts you as follows on all subsequent attempts to format the disk:

Enter current Volume Label for
drive (x):

If you enter an incorrect label, **FORMAT** displays:

Invalid Volume ID Format failure

If you enter the correct volume label (or if you press **ENTER** if the disk has no label), **FORMAT** continues. Before formatting, it displays a warning and asks if you want to proceed.

- If you are formatting a hard disk that can store more than 32 megabytes of data, you need to use **MLPART** and **MLFORMAT** (in addition to **FDISK** and **FORMAT**) to use the entire hard disk. (See the **MLPART** and **MLFORMAT** commands and Appendices C and D.)

Examples

FORMAT A: /S

formats the diskette in Drive A, and transfers the hidden operating system files to the formatted diskette. If the drive is high-capacity, the diskette must be high-density for this command to work. If the drive is standard, the diskette must be double-sided, standard.

FORMAT B: /4

formats a standard, double-sided diskette in a high-capacity Drive B.

FORMAT C: /V/S

formats hard disk Drive C. FORMAT prompts you for a volume label, and transfers the hidden operating system files to the formatted disk.

GOTO

Internal

GOTO *label*

A batch file command that transfers control to the line (in the batch file) following the one specified by *label*.

Parameters

label is a character string. MS-DOS considers only the first eight characters of the string. It ignores the rest. If you omit *label*, execution of the batch file terminates.

Notes and Suggestions

- Use GOTO with the IF command to direct execution to particular subroutines when particular conditions exist. (See the IF command.)
- Using GOTO and IF, you can create a batch file that copies a specified source file to a specified target file only if the target does not already exist. If the target exists, the file pauses so you can halt the copy.
- When a batch file executes, the screen does not display its labels. Therefore, you can use labels to include comments in a batch file.

Examples

```
COPY con chekdest.bat   
REM When executing, substitute the drive for %1  
and the directory for %2   
REM When executing, substitute the source file  
for %3, the target for %4   
%1   
CHDIR %2   
IF NOT EXIST %4 GOTO G   
TYPE %4   
PAUSE   
:G   
COPY %3 %4   
:END   
 
```

is a batch file that checks to see if a file exists before copying it. Suppose Newfile.asm exists in the ROOT directory of Drive B, and you execute the batch file by typing:

```
Chekdest B:\ oldfile.asm newfile.asm 
```

As instructed in the command line, the batch file replaces %1 with B:, making Drive B the current drive. It then replaces %2 with \, making the ROOT directory the current directory. Finally, it replaces %3 with Oldfile.asm and %4 with Newfile.asm.

The batch file checks to see if Newfile.asm exists in the current directory. It does; therefore, the batch file uses the TYPE command to display Newfile.asm. Then it pauses. If you want to copy over Newfile.asm, press the space bar. If not, press .

If Newfile.asm does not exist in the directory, the batch file *goes* to the line following :G. Therefore, it does the copy automatically, without pausing.

GRAFTABL

External

GRAFTABL

Loads the character definitions (dot patterns) for ASCII characters 128-255 into memory.

Parameters

None

Notes and Suggestions

- You must use the GRAFTABL command if you install Tandy's Deluxe Graphics Display Adapter in your computer. Otherwise, when the video display is placed in a graphics mode, only the first 128 characters are defined and usable.
- GRAFTABL remains in effect only until you reboot the computer. Therefore, if you want to continue to use the 128-255 character set in graphics mode after rebooting, you need to rerun GRAFTABL.
- After loading the character table into memory, GRAFTABL prints the message:

GRAPHICS CHARACTERS LOADED

If you execute GRAFTABL again without rebooting, MS-DOS displays the message:

GRAPHICS CHARACTERS ALREADY LOADED
- GRAFTABL increases the size of MS-DOS resident in memory.

Example

```
GRAFTABL 
```

GRAPHICS

External

GRAPHICS *p*type [/R] [/B] [/CR][[/LF]]

Enables you to reproduce a graphics screen in color or in shades of gray on a printer. To reproduce the screen, press **SHIFT** **PRTSC** while in a graphics video mode. Press **SHIFT** **PRTSC** a second time to stop the screen print.

Parameters

*p*type specifies the printer type, and is one of the following printers:

Code	Printer
CGP220	The Tandy CGP-220 color ink jet printer.
STANDARD	Any Tandy printer (other than the CGP-220 or the DMP-110) without DIP switch settings for Tandy and PC printer compatibility.
PCMODE	Tandy printers with a DIP switch set for PC compatibility. Also, use PCMODE for other PC-compatible printers.
TMODE	Tandy printers with a DIP switch set for Tandy compatibility.
DMP110	The Tandy DMP-110 printer.

/R causes GRAPHICS to print black as black and white as white, when you specify a black-and-white printer. If you omit /R, GRAPHICS prints black as white and white as black.

/B causes GRAPHICS to print the background color when you specify a CGP-220 printer. If you omit /B, the background is not printed.

- /CR** causes GRAPHICS to execute an end-of-line character as a carriage return. Otherwise, a carriage return causes both a line feed and a carriage return.
- /LF** causes GRAPHICS to send only a line feed as the end-of-line character. /LF and /CR are mutually exclusive.

Notes and Suggestions

- If you omit *ptype* in the command line, GRAPHICS displays this menu:

```
Tandy Graphics Screen Dump Utility
Version 3.00.00
Copyright 1985 Tandy Corp.,
All rights reserved.
```

```
Enter type of printer
[A] For TANDY CGP-220 printer
[B] For TANDY DMP Standard Resolution
    printer
[C] For PC compatible printer
[D] For PC compatible printer in TANDY
    MODE
[E] For TANDY DMP-110 printer
```

Press the letter that corresponds to your type of printer.
- The /CR and /LF switches only affect printers with DIP switch settings for Tandy and PC compatibility. If you have such a printer, and it does not perform a line feed when required, or if it prints an unwanted blank line between text, use the appropriate /CR or /LF switch to correct the problem.
- To turn off GRAPHICS, reset the computer. If GRAPHICS is turned off, pressing **SHIFT** **PRTSC** causes a text screen to be printed. Pressing **SHIFT** **PRTSC** a second time terminates a current print operation.
- Graphics images are printed in a maximum of eight shades of gray on a black-and-white printer, or in 16 colors on a CGP-220 printer. On black-and-white printers, the image is printed sideways when the video mode is 640x200.
- GRAPHICS increases the size of MS-DOS resident in memory.

- You can invoke GRAPHICS any number of times with different parameters. If the printer type remains the same, only one copy of the program remains in memory.

HSECT

External

HSECT

Formats track and sector information on the hard disk. The HSECT command works the same as the FORMAT HARD DISK utility on your Utilities diskette. After using it, use FDISK and FORMAT to complete the hard disk format process.

Parameters

None

Notes and Suggestions

- After you enter the HSECT command, the following prompt appears:

```
Which hard drive do you  
want to format (C/D)  
?
```

To format an internal hard disk drive, type c ENTER[®]. To format a second internal or an external hard disk drive, type D ENTER[®].

- After you select the disk to format, the screen displays the following warning:

```
All data on drive x will be  
DESTROYED!!  
Do you want to continue (Y/N)  
?
```

If you continue, HSECT erases all data from the hard disk. Therefore, use it only when you are preparing to install an operating system on your hard disk for the first time.

Type N ENTER[®] to exit the formatting procedure and return to the Main Menu, or type Y ENTER[®] to continue.

- If you continue, HSECT displays information about the disk you want to format, like this:

```
Hard drive C is type  x
Number of heads = x
Number of cylinders = x
Is this correct (Y/N)
?
```

At the prompt, verify the hard disk information on the screen. If the displayed information matches your hard disk drive, type Y .

If you made an error in the SETUP program, or if you have a *non-standard* type of hard disk, the information does not match your disk. Type N . Then, answer the prompts that follow with the correct number of heads and cylinders and the interleave factor for your hard disk drive. Also type N if you want to change the interleave factor from the default of 3.

- After you verify or change the hard disk information, HSECT prompts:

```
Do you want to flag defective
tracks (Y/N)
?
```

If your hard disk's Media Error Map shows no defective tracks, type N to begin the formatting procedure.

If the map shows one or more defective tracks, type Y . The following prompt appears on the screen:

```
Enter next head, cylinder pair or
press <enter> to quit.
?
```

As an example, if your Media Error Map lists Head 4, Cylinder 100 and Head 5, Cylinder 100 as defective, type:

```
4,100 
5,100 
```

After you enter all the defective heads and tracks on the map, press **[ENTER]** to begin the formatting procedure.

- Do not interrupt the program while it is formatting the drive. When the format is complete, HSECT displays **Format completed!**

Examples

HSECT **[ENTER]**

loads HSECT from the current drive. HSECT prompts for the drive to format.

B:HSECT **[ENTER]**

loads HSECT from the diskette in Drive B. HSECT prompts for the drive to format.

IF

Internal

IF [NOT] *condition* *command*

Allows conditional execution of commands in batch file processing. When the condition is true, the command is executed. When it is false, the command is ignored.

Parameters

NOT changes the IF command so that the command executes only when the condition is false.

condition is one of the following:

ERRORLEVEL *number* indicates that you want the command to execute only if the program previously executed by **COMMAND** has an exit code of *number* or higher.

string1 == *string2* indicates that you want the command to execute only if *string1* and *string2* are identical after parameter substitution. Strings cannot have embedded separators.

EXIST *filename* indicates that you want the command to execute only if the file specified by *filename* exists.

command is the command that is executed if the *condition* is met.

Notes and Suggestions

- Use IF with the GOTO command to direct execution to particular subroutines when particular conditions exist. (See the GOTO command.)

Suppose you have three programs (Myprog, Samprog, and Salprog). Each requires you to be in a separate directory. To create a batch file to put you in the proper directory, type:

```
COPY con progdir.bat   
REM This file changes the dir for  
myprog, samprog, or salprog   
REM When executing, substitute  
the drive for %1, the program  
name for %2   
%1   
IF %2==myprog GOTO W   
IF %2==samprog GOTO X   
IF %2==salprog GOTO Y   
:W   
CHDIR \USER\MYPROG   
GOTO Z   
:X   
CHDIR \USER\SAMPROG   
GOTO Z   
:Y   
CHDIR \USER\SALPROG   
:Z   
CHDIR   
:END   
 
```

Execute the batch file by entering its file-name, followed by the drive and program name. For example, to change to the correct directory for Myprog, type:

```
progdir B: myprog 
```

The batch file substitutes B: for %1, making Drive B your current drive. Then, it substitutes Myprog for %2. It goes to label :W, and changes the directory to \USER\MYPROG. Then it goes to :Z. The line following :Z causes CHDIR to display the current directory to verify that it is correct.

Examples

```
IF NOT EXIST myfile ECHO Can't find file 
```

displays Can't find file if the file Myfile doesn't exist.

```
IF EXIST all.1st GOTO G 
```

sends program execution to the line after :G if All.1st exists.

JOIN

External

JOIN *drive pathname /D*

Links the ROOT directory of the specified *drive* to a different *pathname*.

Parameters

<i>drive</i>	is the drive to be joined.
<i>pathname</i>	is the path (including the drive) to which <i>drive</i> is joined.
<i>/D</i>	turns off the effects of a previous JOIN command.

Notes and Suggestions

- The JOIN command removes the distinction that physical drives are separately addressable by drive letter. Using JOIN, you can always refer to all the directories on a specific drive by one pathname. If the path already exists, it is not usable while JOIN is in effect.
- If the pathname does not exist, MS-DOS tries to create a directory with that pathname. The path must be empty. After you issue the JOIN command, the first drive name becomes invalid. If you try to use that drive name, MS-DOS displays the error message *Invalid drive*.
- You can join a drive only at the ROOT. For example, you can use the command:

```
JOIN D: C:\MEMOS
```

but you cannot use:

```
JOIN D: C:\MEMOS\JUNE
```


- To undo the effects of a JOIN command, use this format:

`JOIN drive /D`

drive represents the drive you wish to deselect. The /D switch turns off the effects of the JOIN command.

- To display the current JOIN status, type:

`JOIN`

Examples:

`JOIN B: A:\HARDWARE`

joins the ROOT directory of Drive B to the pathname A:\HARDWARE. Now, when you specify A:\HARDWARE, MS-DOS assumes you mean the ROOT directory of Drive B.

KEYBxx

External

KEYBFR [/US]

KEYBGR [/US]

KEYBIT [/US]

KEYBSP [/US]

KEYBUK [/US]

Replaces the current ROM BIOS keyboard program with an international keyboard program.

Parameters

/US tells KEYBxx that character scan codes are to be converted to US scan codes.

Notes and Suggestions

- The following are the available keyboard programs:

KEYBFR	France
KEYBGR	Germany
KEYBIT	Italy
KEYBSP	Spain
KEYBUK	United Kingdom
- The /US parameter is required to operate some application programs that are configured for a US keyboard.
- To return to the US keyboard layout from a KEYBxx program, press **CTRL** **ALT** **F1**. To select the resident KEYBxx program, press **CTRL** **ALT** **F2**.
- Although you can have more than one KEYBxx program resident in memory at one time, you are only able to operate in the last loaded version.

LABEL

External

LABEL [*drive*][*label*]

Lets you create, change, or delete a volume label for the current or specified *drive*.

Parameters

drive is the disk drive for which you want to create or modify a label. If you omit the *drive*, the command applies the label to the current drive.

label is the volume name. It can be a maximum of 11 characters. If you omit the *label*, the command prompts you for a label.

Notes and Suggestions

- The volume *label* can include spaces, but not tabs. Do not use the foolowing characters in the *label*:

* ? / \ | . , ; : + = < > []

- If you omit the *label*, the command prompts:

```
Volume in drive x is
    current label
Volume label (11 characters,
ENTER for none)?
```

or

```
Volume in drive x has no label
Volume label (11 characters,
ENTER for none)?
```

Either enter the label you want, or simply press **ENTER** to indicate that you want no label. If a label exists, and you press only **ENTER**, the command prompts:

```
Delete current volume label
(Y/N)?
```

Type Y to delete the current label or N to keep the label.

- You cannot use LABEL over a network.

Example

```
LABEL A:MYDISK 
```

assigns the diskette in Drive A the label MYDISK.

LF

External

LF

Suppresses the line feed after a carriage return in output to printer 1, 2, or 3.

Parameters

None

Notes and Suggestions

- Some printers automatically generate a line feed after they receive a carriage return. With some programs, this causes an extra line to print between each line of characters. LF suppresses the line feed sent to the printer by the software.
- LF suppresses only those line feed codes are that immediately follow a carriage return.
- LF supports the **MODE LFOFF**, **MODE LFON**, **MODE DMP**, and **MODE DWP** commands. **LPDRVR.SYS** also supports these **MODE** commands.

MKDIR

Internal

MKDIR *pathname*

MD *pathname*

Creates (makes) a directory. Use the MKDIR command to create a hierarchical directory structure.

Parameters

pathname is the name you want to give the subdirectory you are creating. Unless the *pathname* specifies otherwise, MKDIR creates the new directory as a subdirectory of the current directory.

Notes and Suggestions

- Using MKDIR, you can better organize your files. For example, you can group files according to user or use.
- You can also use MKDIR to create a directory in which to put your external commands.

Examples

```
MKDIR \BIN 
```

creates a directory named BIN as a subdirectory of the ROOT directory. You can use the BIN directory to contain your external commands. Use COPY to copy your command files into this directory; then, use PATH to tell MS-DOS where to search for the commands.

```
MKDIR \USER 
```

creates the subdirectory \USER in your ROOT directory.

```
MD \USER\JOE 
```

creates the subdirectory JOE in your \USER directory.

MLFORMAT

External

MLFORMAT *drive*

Formats a DOS2 partition created previously using MLPART.COM (and accessed via the installable MLPART.SYS device driver). You must format the partition to use it.

Parameters

drive is the **logical** drive letter that refers to the DOS2 partition to format. MS-DOS assigns and displays the letter when it installs the MLPART.SYS device driver (loads it from disk into memory). It displays the letter in the following format:

```
Tandy MLPART version xx.xx.xx
  split disk drive
Installing additional drive on
  physical drive letter as
  Drive logical drive letter
--INSTALLED--
```

Notes and Suggestions

- See the MLPART command for a synopsis of the entire hard disk initialization procedure. See Appendices C and D for more information on installing MLPART.SYS.

Examples

```
MLFORMAT E: 
```

formats logical drive E.

MLPART

External

MLPART

Lets you create multiple non-bootable (DOS2) partitions on a hard disk, after you create a DOS partition. The DOS2 partitions cannot be *active* (bootable).

This command (MLPART.COM) is used in conjunction with the MLFORMAT command, which formats DOS2 partitions, and the MLPART.SYS device driver, which enables you to access DOS2 partitions. It is intended for use with hard disks that have a capacity of 32 or more megabytes. The MS-DOS partition must start in the first 32 megabytes of a hard disk, but DOS2 partitions can start above the 32-megabyte boundary. Therefore, DOS2 partitions enable you to make use of the remainder of the hard disk's space.

Here is a synopsis of the entire procedure for partitioning Drive C and making it bootable. If you have performed some of the steps before, do not repeat them, unless you first back up the information on your hard disk. The Format Hard Disk utility, HSECT, FORMAT, and MLFORMAT all erase information on the disk. Normally, you need to do the procedure only once.

1. Use the Format Hard Disk utility (on the Utilities diskette) or the HSECT command to format the hard disk.
2. Use FDISK to create a DOS partition on Drive C.
3. Use FORMAT to format the DOS partition and make it bootable.
4. Copy all files from your MS-DOS system diskette to Drive C.
5. Use MLPART.COM to create the non-bootable, DOS2 partition(s) on Drive C. (MLPART.COM is on the Supplemental Programs diskette.)
6. Copy MLPART.SYS from the Supplemental Programs diskette to Drive C.
7. Include the `DEVICE = MLPART.SYS C:` command in a CONFIG.SYS file on Drive C.
8. Reboot the computer from Drive C to install the MLPART.SYS device driver (load it into memory).

- 9. Use **MLFORMAT** to format the DOS2 partition(s).

See the specific commands and Appendices C and D for more information.

Parameters

None

Notes and Suggestions

- When you enter the **MLPART** command, the screen displays the following menu:


```
1. Create DOS2 Partition
2. Delete DOS2 Partition
3. Display Partition Data
4. Select Next Hard Disk Drive
5. Select Previous Hard Disk
   Drive
Enter Selection -->
Press ESC to exit to MSDOS
```
- You can place DOS2 partitions in any order on the disk. When you select the **Create DOS2 Partition** option, **MLPART** displays the number of cylinders available on the disk and the starting cylinder of the available space. It asks you to enter the partition size (in cylinders) and the starting cylinder number.

You can have a maximum of four partitions (including the MS-DOS partition) on a disk, and each can be a maximum of 32 megabytes.
- The remaining options are similar to their **FDISK** counterparts. See the **FDISK** command for more information.

Examples

MLPART

loads MLPART from the current drive, and displays the MLPART menu. To create one or more DOS2 partitions (assuming you have a DOS partition), press . The screen displays the status of Drive C in this format:

Partition	Status	Type	Start	End	Size
1	N	DOS	0	49	50

The status column refers to *active* or *nonactive*. An active partition is booted directly from the drive when you start the system. The other columns indicate that there is one partition, beginning at Cylinder 0 and ending at Cylinder 49, for a total size of 50 cylinders, and that the partition contains a disk operating system.

The screen asks for the information MLPART needs to create your first DOS2 partition. After MLPART creates the partition, you can select Menu Option 1 again to create more DOS2 partitions.

You can choose Next Hard Disk Drive if you want to create multiple partitions on Drive D. After creating the partitions, you can select Display Partition Data to verify the size and location of each partition.

MODE

External

Sets parameters for a video monitor, a line printer, a communication interface, and the CPU.

MODE [*characters*][*shift*[T]]

Shifts the video screen left or right. *characters* is the desired line width (40 or 80). *shift* can be R (right) or L (left). Using T produces a video test screen for evaluating the shift.

MODE *linefeed*

Sets the printer linefeed off or on. *linefeed* can be LFOFF or LFON.

MODE *printer*

Sets the printer type. *printer* can be DMP (dot matrix), DWP (daisy wheel), or NL (reset).

MODE *trans*

Sets MS-DOS to properly translate video characters for Tandy printers during screen print procedures. *trans* can be:

DMPXLAT	— for Tandy DMP printers
DWPXLAT	— for Tandy DWPII printers
DWP10	— for Tandy DWPIIB, DWP410, or DWP510 printers with 10-pitch settings
DWP12	— for Tandy DWPIIB, DWP410, or DWP510 printers with 12-pitch settings
NOXLAT	— returns translations to the default setting (no translation)

To use MODE *trans*, you must first load LPDRV.RSYS.

MODE [*video*][*characters*]

Sets the video mode and characters-per-line. *video* can be BW (black and white), CO (color), or MONO (changes to the monochrome adapter with 80 columns and 25 rows). *characters* can be 40 or 80.

**MODE COM*number*: [*baud*][*parity*][*databits*][*stopbits*]
[P]**

Sets the RS232 communication parameters:

number is the RS232 port to set, either 1 or 2.

baud can be a baud rate of 110, 150, 300, 600, 1200, 2400, 4800, 9600, or 1200/75. If the baud rate is set at 1200/75, MODE initializes the international parallel/serial adapter: The transmit baud rate is set to 75, and the receive baud rate is set to 1200.

parity can be N (no parity), O (odd parity), or E (even parity).

databits can be either 7 or 8 databits. The default is 7.

stopbits can be either 1 or 2 stopbits. The default is 1.

P sets the specified serial port for continuous timeout retries. If you want continuous timeout retries on parallel output redirected to a RS232 port, use P with the MODE COM*number*: first.

MODE SLOW | FAST

Sets the CPU speed to either 4 megahertz (SLOW) or 8 megahertz (FAST). This enables you to use your computer with a variety of application programs and options, regardless of the speed they require. If you do not use this command, the CPU operates at 8 megahertz.

MODE LPT*number*: *characters* [/type] [,P]

Sets the characters-per-line for the printer specified by *number*. /*type* can be /DMP (dot matrix) or /PC (PC compatible). The default for /*type* is /DMP. There should not be a space before /*type* in the command line. *number* can be 1, 2, or 3 for your computer's three parallel ports. *characters* can be 80 or 132. P means the driver continuously retries to output on timeouts. Return to normal functions by issuing the MODE LPT command without the P parameter.

MODE LPT*number*:*timeout*

Sets the timeout delay for the printer specified by *number*. *number* can be 1, 2, or 3 for your computer's three parallel ports. *timeout* can be LONG (2 minutes) or SHORT (45 seconds).

MODE LPT $number$ = COM $serial$

Redirects printer output from the specified parallel printer port (*number*) to the specified RS232 channel (*serial*). *number* can be 1, 2, or 3 for your computer's three parallel ports. *serial* can be 1 or 2. Initialize the selected RS232 channel using MODE COM-*number*: before redirecting printer output. To redirect output back to the parallel printer port, enter any MODE LPT command, specify the number of the parallel port from which you originally redirected output.

Examples

```
MODE BW 40
```

sets the video monitor to black and white with 40 characters per line.

```
MODE LFOFF
```

causes the line printer driver to suppress line feeds following a carriage return.

```
MODE LPT1:80/DMP
```

sets the line printer column width to 80 characters, and initializes MS-DOS for a Tandy dot matrix printer.

```
MODE LPT1:=COM1
```

redirects output from the first parallel printer port to the first RS232 communications port.

```
MODE COM1: 300 N 8 1
```

sets the first RS232 interface to 300 baud, no parity, 8 databits, and 1 stop bit.

```
MODE LPT1:SHORT
```

sets the timeout delay for Printer 1 to 45 seconds.

```
MODE DWP
```

sets printer parameters to use a Tandy daisy wheel printer.

```
MODE 80 L T
```

shifts the video display one character left, and produces test characters. Continue to press **[L]** for left or **[R]** for right until the display is satisfactory.

MORE

External

MORE

MORE is a filter that reads from standard input (such as a command from your keyboard) and displays one screen of information at a time. It then pauses and displays the message -MORE- at the bottom of your screen.

Press to display another screen of information. This process continues until all the input data is read.

Examples

```
TYPE myfiles.com|MORE 
```

displays the file Myfiles.com one screen at a time. MS-DOS displays the message -MORE- at the bottom of each screen. Press to continue.

```
TYPE B:acctspay.dat|MORE 
```

displays the file Acctspay.dat from the home directory of Drive B, one screen at a time. MS-DOS displays the message -MORE- at the bottom of each screen. Press to continue.

PATCH

External

PATCH *pathname,address,data1,data2*

Lets you make minor corrections in any disk file, provided that:

- You know the location of the bytes you want to change.
- You want to replace one list of hexadecimal values with another list of the same length.

Parameters

<i>pathname</i>	is the file you want to change.
<i>address</i>	is the starting byte of the data to be changed (in hexadecimal).
<i>data1</i>	is a list of the hexadecimal values to be changed.
<i>data2</i>	is a list of the hexadecimal data values to replace <i>data1</i> .

Notes and Suggestions

- In addition to using PATCH to change your own programs, you can use it to implement minor program changes from Tandy. If a Tandy program ever requires such a change, Tandy will distribute a document that specifies the parameters you need for the patch.
- If *data1* is not at *address*, PATCH generates an error message and quits.

Example

```
PATCH B:BASIC.EXE,65234.0001020304,1011121314
```

```
ENTER
```

patches the file BASIC.EXE, located on Drive B. The command changes Bytes 65234 through 65238 to the hexadecimal values 10, 11, 12, 13, and 14—assuming that they are originally 00, 01, 02, 03, and 04.

PATH

Internal

PATH [;] [*pathname*] [;*pathname*...]

Sets a command path, which tells MS-DOS the directories or drives in which to search for external commands. You can also use PATH to remind you of the current path.

Unless you reset the path or turn off the system, MS-DOS searches the specified path(s) each time you use an external command.

Parameters

pathname can specify either a directory or an entire drive.

If you specify PATH; (PATH followed by a semi-colon), MS-DOS searches the current directory only.

Notes and Suggestions

- MS-DOS always searches the current directory before searching the directories specified by PATH.
- If you don't include a pathname, PATH displays the current path setting. This serves to remind you which path MS-DOS is searching. If no path is set (MS-DOS is searching only the current directory), PATH displays the message No path.
- PATH gives you the option of using commands that are not in the current directory. Suppose the ROOT directory in Drive A contains so many files it is difficult to use efficiently. To save space, create the directory A:BIN (using MKDIR) and put your commands there (using COPY). Then remove your commands from A: (using DEL).

FORMAT is an external command. Therefore, to format a disk in Drive B, you must do either of the following:

- Use CHDIR to make A:BIN your current directory.
- Use PATH to make MS-DOS search A:BIN for external commands. You can specify A:BIN only, or you can specify all directories on Drive A, separating them with semicolons. To specify A:BIN only, type:

```
PATH A:\BIN 
```

Type PATH to verify the path setting.

Examples

```
PATH \BIN\USER\JOE 
```

tells MS-DOS to search \BIN\USER\JOE in the current drive for external commands (after it searches the current directory).

```
PATH BIN\USER\JOE;\BIN\USER\SUE;\BIN\DEV 
```

tells MS-DOS to search the directories specified by the above pathnames for external commands. MS-DOS searches the current directory and then those directories listed in the command, in the order in which they are listed.

```
PATH;
```

tells MS-DOS to search the current directory only.

```
PATH 
```

displays the current path setting.

PAUSE

Internal

PAUSE [*message*]

Suspends execution of the currently executing batch file. You can either exit the file or continue. You can use PAUSE only within a batch file.

Parameters

message is a message you want the screen to display when the file pauses. The file displays the message, followed by:

Strike a key when ready...

Notes and Suggestions

- When MS-DOS encounters the PAUSE command, it stops and displays the Strike a key message.

At this time, you can press the space bar to continue execution, or you can press **CTRL** **C** to display:

Terminate batch job (Y/N)?

If you press **Y**, execution ceases and control returns to MS-DOS. If you press **N**, execution continues.
- During the execution of the batch file, you might need to perform some action before executing the next command. For example, you might need to change disks or make sure your printer is ready. Whenever this is the case, include a PAUSE command where necessary.

For example, suppose you want to list two files that are on different disks. Create a batch file by typing:

```
COPY con typfiles.bat 
REM This file types the files
rental.dat and sales.dat 
REM It is called typfiles.bat

TYPE B:rental.dat 
PAUSE 
TYPE B:sales.dat 
 
```

Typfiles.bat displays the file Rental.dat, then pauses and displays the Strike a key when ready... message. Change disks and press the space bar to display Sales.dat.

- Use PAUSE whenever you want an option to cease execution of the batch file.

Suppose you create a batch file that combines all current directory files that have the extension .1st into the file All.1st. If the file All.1st already exists, the copy will destroy the original contents of All.1st.

To avoid destroying All.1st, create this batch file:

```
copy con comblst.bat 
REM This file combines *.1st
files into all.1st 
ECHO OFF 
IF EXIST all.1st ECHO all.1st
already exists. Press [CTRL-C] to
end, or 
PAUSE 
ECHO ON 
COPY *.1st all.1st 
 
```

If All.1st exists, and you execute the batch file, MS-DOS pauses and displays the message:

```
all.1st already exists. Press  
[CTRL-C] to end or  
Strike a key when ready...
```

If All.1st does not exist, MS-DOS pauses and displays only:

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

Examples

```
ECHO Press CTRL-C if unsure or   
PAUSE 
```

pauses execution and displays the message:

```
Press CTRL-C if unsure or Strike a key when ready...
```

```
ECHO Insert disk to check in Drive B --   
PAUSE 
```

pauses execution and displays the message:

```
Insert disk to check in Drive B--  
Strike a key when ready...
```

PRINT

External

PRINT [*pathname*] [/D:*device*] [/B:*size*] [/Q:*value*]
[/C...] [/P...] [/T]

Lets you queue up files to print on a line printer while you are processing other commands. This is called *background printing*.

Parameters

<i>pathname</i>	specifies the file(s) to print. The following switches are allowed the first time you use PRINT after starting or resetting MS-DOS:
/D: <i>device</i>	(device) specifies the printer device. If you omit the <i>device</i> , PRINT uses LPT1. If you don't use this parameter the first time you use PRINT, the command prompts you for the device.
/B: <i>size</i>	(buffer) sets the size, in bytes, of the internal buffer. Increasing the size of the buffer speeds up PRINT operations.
/Q: <i>value</i>	(queue) specifies the number of files allowed in the print queue if you want more than 10, which is the default number. The minimum is 4; the maximum is 32.
/C	(cancel) deletes from the print queue the file immediately preceding /C in the command line and all files following /C.
/P	(print) adds to the print queue the file immediately preceding /P in the command line and all files following /P.
/T	(terminate) deletes all files in the print queue. A message to this effect is printed.

Notes and Suggestions

- When used without options, PRINT displays the contents of the print queue on the screen without affecting the queue.

- Use PRINT only if you have a line printer attached and ready.
- Each print queue entry can contain a maximum of 64 characters, including the drive name. Therefore, you might need to change directories if you want to print files in deep subdirectories.

Examples

```
PRINT temp1.tst temp2.tst temp3.tst 
```

stores the three files indicated in the print queue. If your printer is connected and ready, background printing begins.

```
PRINT /T 
```

empties the print queue.

```
PRINT A:temp1.tst /C A:temp2.tst A:temp3.tst  

```

removes the three files indicated from the print queue.

```
PRINT temp1.tst /C temp2.tst /P temp3.tst 
```

removes Temp1.tst from the queue, and adds Temp2.tst and Temp3.tst to the queue.

PROMPT

Internal

PROMPT [*text*]

Changes the MS-DOS system prompt to *text*.

Parameters

text is a string of characters to set as the prompt. It can be any of the following:

- A string of characters, such as your name.
- A special prompt in the format *\$character*, in which *character* is one of those in the chart below.
- A combination of characters, strings, and special prompt(s).

In the prompt text you must precede the following symbols with the \$ character.

Precede this character with \$

To get this prompt

t	<i>current time</i>
d	<i>current date</i>
p	<i>current directory</i>
v	<i>MS-DOS version number</i>
n	<i>current drive specification</i>
g	>
l	<
b	
—	carriage return and line feed
q	=
h	backspace (to end of preceding line)
e	the 'ESC' character
\$	\$

Notes and Suggestions

- If you omit *text*, MS-DOS sets the current drive specification as the prompt.
- By setting the current directory as the prompt, you need not enter the CHDIR command to remind you which directory you are in. To set the directory as the prompt, type:

```
PROMPT $p 
```

- By setting the time as the prompt, you can check the time without entering the TIME command. In this way, you can also time the execution of commands and programs. To set the time as the prompt, type:

```
PROMPT $t 
```

- You can use \$__ to insert a carriage return and line feed in your prompt:

```
PROMPT TIME = $t$__DATE = $d  

```

In this case, the new system prompt is similar to the following:

```
TIME = 0:07:04.07  
DATE = Thu 11-15-1985
```

- Because your computer has an ANSI escape sequence driver (ANSI.SYS), you can use escape sequences in your prompts, if the ANSI device driver is configured into your system. (See Appendix C.) For example:

```
PROMPT $e[7m$n:$e[m 
```

sets the prompts in reverse video mode, and returns to video mode for other text.

Examples

```
PROMPT $n$g 
```

sets the current drive, followed by a greater-than sign (>) as the prompt. For example, when you change to Drive B, the prompt changes to B>. When you receive your system, PROMPT is set to \$n\$g.

```
PROMPT A$g 
```

sets the prompt A>. Regardless of which drive you are in, this is the prompt.

```
PROMPT $p 
```

sets the current directory, including the current drive, as the prompt.

Note: As you can see from the A\$g example, some prompts hinder rather than help. Use PROMPT carefully.

RECOVER

External

RECOVER [*drive* | *pathname*]

Recovers a file that contains bad sectors, or recovers all files on a disk that contains bad sectors in its directory.

Parameters

<i>pathname</i>	specifies the file to recover. You must include the filename part of the pathname.
<i>drive</i>	specifies the disk to recover.

Notes and Suggestions

- In recovering files that contain bad sectors, MS-DOS reads the file sector by sector. It marks the bad sectors in a system table so that the data is never again allocated to them.
- In recovering files that contain bad sectors, MS-DOS scans the disk File Allocation Table (FAT) for chains of allocation units. It creates a new root directory for each chain. Use the RECOVER *drive* command only if the disk's directory is unusable.
- If there is not enough room in the root directory, RECOVER displays a message to this effect. It then stores information about the extra files in the File Allocation Table. When there is enough room, you can use RECOVER again to regain the files.
- If you have trouble storing and manipulating information, the disk may have a flawed sector. Running CHKDSK (check disk) should indicate whether this is the case. If the file that contains the flawed sector is a text file, recover the file. This saves all information except that in the flawed sector. Re-enter the lost information. If the file is a data file, recovering the file may or may not help.

- If you are a beginner, you might prefer reentering all information rather than trying to recover a file.
- You cannot use RECOVER over a network.

Examples

```
RECOVER \USER\SAM\pamphlet.txt 
```

recovers the file \USER\SAM\Pamphlet.txt that is on the current disk.

```
RECOVER oldbook.txt 
```

recovers the file Oldbook.txt that is in the current directory.

```
RECOVER B: 
```

recovers the disk in Drive B if the bad sectors are in the directory.

REM (Remark)

Internal

REM [*remark*]

Lets you include the specified remark in a batch file. You can use this command only within a batch file.

Parameters

remark is a line that a batch file displays during execution. The line is displayed only if ECHO is on. The space, tab, and comma are the only separators allowed in a remark.

Notes and Suggestions

- You can use REM to remind you of the file's name and use, keep track of what a particular command is doing, or include warnings in the file.
- You can also use REM, without a comment, to add space for better readability.

Examples

```
COPY CON newdisk.bat   
REM This file checks new disks   
REM It is called Newdisk.bat   
FORMAT B: /S   
CHKDSK B: 
```

creates a batch file that formats and checks new disks after reminding you of the file's name and purpose.

REN (Rename)

Internal

REN *pathname filename*

Changes the name of the file specified by *pathname* to *filename*.

Parameters

pathname is the file to be renamed.

filename is the new name for the file.

Notes and Suggestions

- If you include a wild card in the *pathname*, MS-DOS renames all files in the specified directory that match the *pathname*. For example, if you type the following:

```
REN *.1st *.prn 
```

MS-DOS changes only the extension of all .1st files that are in the current directory. The new extension is .prn. The file Suefile.1st, for example, becomes Suefile.prn.

- If you include a wild card in the *filename*, MS-DOS leaves the corresponding characters as they were in the *pathname*. For example, if you type the following:

```
REN newfile old???? 
```

MS-DOS replaces the ? wild cards with the characters that occupied the same position in the original name. It changes Newfile to Oldfile.

Examples

```
REN B:\USER\gl1.dat gl2.dat 
```

changes the name of the Gl1.dat file that is in B:\USER to Gl2.dat.

```
REN B:mufile ?y???? 
```

changes the name of Mufile, a file in the home directory in Drive B, to Myfile.

REPLACE

External

REPLACE *source pathname* [*target pathname*] [/A]
[/D] [/P] [/R] [/S] [/W]

Updates previous versions of files by replacing them with new versions; adds new files to an existing directory.

Parameters

*source
pathname* is the drive or directory that contains the replacement files. It can also be a single file or a wild card filename.

*target
pathname* is the drive or directory that contains the files you want to replace.

/A adds files that exist in the source directory—but **not** in the target directory—to the target directory.

/D replaces the target files only if the source files are newer. You cannot use this switch and /A at the same time.

/P causes MS-DOS to prompt you so that you can verify whether or not to add each source file (or replace each target file). The screen displays:

Replace (*filename*) ? (Y/N)

/R replaces read-only files, as well as unprotected files. If you try to replace a read-only file without specifying /R, an error results, and the REPLACE process stops.

/S causes REPLACE to search all subdirectories of the target directory while it replaces matching files. You cannot use this switch and /A at the same time.

/W causes REPLACE to wait for you to press a key before it replaces the files. If you omit this switch, REPLACE automatically replaces or adds files.

Notes and Suggestions

- **REPLACE performs two functions:**

By default, it replaces files in the target directory with files in the source directory that have the same name.

If you specify the /A switch, REPLACE adds files that exist in the source directory—but not in the target directory—to the target directory.
- As it replaces or adds files, REPLACE displays the filenames on the screen. When finished, it displays:


```
nnn file(s) added/replaced
```


or


```
No files added/replaced
```
- You cannot use REPLACE to update hidden or system files.

Examples

Suppose Drive C contains several files of clients' names and phone numbers, each in a separate directory but with the same filename, PHONES.CLI. To replace these files with the latest version, which exists in a file on Drive A, type:

```
REPLACE A:\phones.cli C:\ /S 
```

The preceding command replaces every file on Drive C that is named Phones.cli with the file Phones.cli from the ROOT directory of Drive A.

Suppose you want to add some new printer device drivers to a directory called C:\TOOLS, which already contains several printer driver files for a word processor. To do this, type:

```
REPLACE A:*.prd C:\TOOLS /A 
```

The preceding command adds any files from the current directory of Drive A that have the extension .prd—and that do not already exist in the \TOOLS directory of Drive C—to C:\TOOLS.

RESTORE

External

RESTORE *source drive target pathname* [/S] [/P]
[/A:date] [/B:date] [/E:time] [/L:time]

Restores one or more files previously backed up using the BACKUP command. You can restore files from one type of media to another. For example, you can restore:

- from floppy disk to hard disk
- from hard disk to floppy disk
- from floppy disk to floppy disk
- from hard disk to hard disk

The primary use of RESTORE is to return files from floppy diskettes to a hard disk.

Parameters

<i>source drive</i>	is the drive containing the backed up file(s).
<i>target pathname</i>	is the directory to which you want to restore the file(s).
<i>pathname</i>	specifies the disk directories and/or file(s) you want to restore.
/S	(subdirectories) causes the command to restore the specified directory and all its subdirectories.
/P	(prompt) causes the command to prompt you for permission to restore any hidden or read-only files that match <i>pathname</i> and any files changed since the last backup.
/B:date	(before) restores only those files last modified on or before the specified <i>date</i> .
/A:date	(after) restores only those files last modified on or after the specified <i>date</i> .
/E:time	(earlier than) restores only those files last modified at or earlier than the specified <i>time</i> .
/L:time	(later than) restores only those files last modified at or later than the specified <i>time</i> .

- /M** (modified) restores only those files changed since the last backup.
- /N** (non-existent) restores only those files that no longer exist on the target disk.

Notes and Suggestions

- This command is compatible with the IBM BACKUP command, except for the /A, /B, /E, /L, /M, and /N switches. You might lose compatibility if you use any of those switches.
- Use the /P option if you are restoring files backed up with a version of MS-DOS that is older than the one on the target disk. When the command asks if you want to restore the system files, press **[N]**, for “no.” In doing this, you avoid writing over a later version of your system with an earlier version.

Examples

```
RESTORE A: C:\ /S [ENTER]
```

restores from Drive A all files backed up from all directories on Drive C.

```
RESTORE A: C:SALES *.dat /P [ENTER]
```

restores all files from Drive A that have the extension .dat, previously backed up from the SALES directory of Drive C.

RMDIR (Remove Directory)

Internal

RMDIR *pathname*

RD *pathname*

Removes the specified subdirectory from a disk.

Parameters

pathname is the name of the subdirectory to be removed.

Notes and Suggestions

- Whenever you delete or copy all of a directory's files, you might want to remove the directory. Doing so saves disk space.
- A directory must be empty (must contain no files or subdirectories) before you can delete it. Copy any files you want to keep to another directory before using DEL or ERASE to remove them from the directory you want to delete.
- The subdirectory you are removing cannot be the current directory. Therefore, you might need to change directories before using RMDIR.

Examples

```
RMDIR \BIN\USER\JIM 
```

removes the subdirectory \BIN\USER\JIM from the current disk.

```
RMDIR MEMOS 
```

removes the subdirectory MEMOS from the current directory.

```
RMDIR B:MEMOS
```

removes the subdirectory MEMOS from the home directory in Drive B.

```
COPY drugstor.dat B:\USER\SUE   
ERASE drugstor.dat   
CHDIR..   
RMDIR USER\ANN\ACCTS 
```

are commands that remove the current directory \USER\ANN\ACCTS, which contains the file Drugstor.dat, from the current drive. The first command copies Drugstor.dat to another directory, which is on Drive B. The second command erases Drugstor.dat from the current directory. The third command changes directories, and the last command erases the original directory, \USER\ANN\ACCTS.

SELECT

External

SELECT *country* [*keyboard*[/US]]

Changes the current country code, or creates an internationally configured backup MS-DOS diskette.

Parameters

country is the country code that MS-DOS uses to select the date and time format, the currency symbol and the decimal separator. The following table shows the available country codes:

Country	Country Code	Keyboard Code
Australia	061	US
Belgium	032	FR
Canadian-French	002	US
Denmark	045	*
Finland	358	*
France	033	FR
Germany	049	GR
Italy	039	IT
Israel	972	US
Middle East	785	US
Netherlands	031	UK
Norway	047	*
Portugal	351	SP
Spain	034	SP
Sweden	046	*
Switzerland	041	*
United Kingdom	044	UK
United States	001	US

An asterisk (*) denotes keyboard programs provided separately.

<i>keyboard</i>	specifies a 2-character identifier of the keyboard layout. You can choose any of the keyboard codes from the previous table, or examine the directory of your MS-DOS diskette for additional KEYBxx.COM commands. Be sure that your MS-DOS system diskette is located in Drive A and that it contains the KEYBxx.COM file that matches your keyboard selection.
/US	specifies US scan codes. You might need to set this parameter to operate some application programs that are configured for a US keyboard. You can include /US only if you also include <i>keyboard</i> .

Notes and Suggestions

- If you specify only *country*, MS-DOS only changes the current configuration to the new country code. If you also specify *keyboard*, with or without /US, the system creates an internationally configured MS-DOS diskette. When you make an international diskette, the MS-DOS system diskette must be in Drive A.
- You must reboot the computer with the newly created MS-DOS diskette for the new keyboard layout and country code to change.
- SELECT creates **new** AUTOEXEC.BAT and CONFIG.SYS files on the backup diskette. It does not copy the original versions of the files.

Examples

```
SELECT 044 UK ENTER
```

selects the United Kingdom date, time, currency symbol, and decimal separator, and the United Kingdom keyboard layout. A new MS-DOS system diskette is created with these new settings.

SELECT 49 GR /US

selects the German date, time, currency symbol, and decimal separator and causes character scan codes to convert to US scan codes as required by some keyboard layout and application programs. A new MS-DOS system diskette is created with these new settings.

SELECT 33

causes the current date and time format, currency symbol, and decimal separator to change to French formats.

SET

Internal

SET `[[string1]=[string2]]`

Sets one string value in the environment equal to another string, for later use in programs you have written.

Parameters

string1 is the string you wish to replace.

string2 is the substitute string.

Notes and Suggestions

- This command is only useful in programs you have written.
- MS-DOS reserves a part of memory for *environment* strings. If you use a name that already exists in the environment, it is replaced by the new string.
- Omitting the second string causes the association to be removed from the environment.
- Typing SET with no parameters displays the current environment settings.
- An application program can get a listing of all environment values by examining the environment. Environments are passed in the Program Segment Prefix. (See information about MS-DOS control blocks and work areas in the *MS-DOS Programmer's Reference Manual*.)
- You can also use SET in batch file processing. Define replaceable parameters with names instead of numbers. For instance, if a batch file contains the statement `CHDIR %PATH-NAME%`, you can set the name for MS-DOS to use with the following SET command:

```
SET pathname=C:/SALES ENTER
```

Examples

SET drive=B:

sets B: to replace the parameter *drive* in later programs.

SET drive=

terminates the current setting for the *drive* parameter.

SET TTY=VT52

sets your TTY (console) value to VT52 until you use another SET command to change it.

SET

displays the current contents of the environment.

SETUP

External

SETUP

Initializes the system configuration parameters retained by your computer's non-volatile memory (CMOS).

Parameters

None

Notes and Suggestions

- Execute SETUP when:
 - You change the hardware configuration of your computer (add or subtract memory, disk drives, video adapters).
 - You wish to change the date and time parameters retained by your computer.
 - You replace the CMOS RAM battery.
- When you execute SETUP, screen prompts display the current configuration of each setting, and ask if they are correct. If the setting is correct, answer Y . If a setting is not correct, answer N . If you answer N , SETUP ask for the new setting. Some settings require only that you type Y or N . Other settings require specific information.
- Configuration information includes:
 - The current time and date—the time and date at the moment you answer the prompt.
 - Diskette Drive A and B types—360K or 1.2M. (Specify 360K for a 3½-inch drive.)
 - Fixed Disk Drives C and D types—can be in the range 1 to 15, for instance TYPE 2. If you do not know your hard disk type, check your hard disk manual or look inside your computer at the hard disk label placed on top of the drive housing.

- System base memory—either 512 or 640 (512,000 bytes or 640,000 bytes).
 - Amount of expansion memory—can be up to 16000 (16,000,000 bytes).
 - The primary video adapter type—color, monochrome, or EGA.
- When SETUP is complete, you must reset and reboot your computer.

SHARE

External

SHARE [/F:space][/L:locks]

Installs file sharing and locking for active networking.

Parameters

- | | |
|-----------------|--|
| <i>/F:space</i> | allocates file space (in bytes) for record file-sharing information. Each open file needs the length of the full filename plus 11 bytes. The average pathname is 20 bytes. The default value for <i>space</i> is 2048 bytes. |
| <i>/L:locks</i> | allocates the number of locks allowed in record filesharing. The default for <i>locks</i> is 20. |

Notes and Suggestions

- Once you use SHARE in an MS-DOS session, MS-DOS checks all read and write requests.
- MS-DOS uses SHARE only when networking is active. Include the SHARE in the AUTO-EXEC.BAT file if you wish to install shared files at startup.

Example

```
SHARE /F:4096 /L:30 ENTER
```

doubles the default number of bytes for the storage of filesharing information, and increases the number of allowable locks to 30.

SHIFT

Internal

SHIFT

Lets you use more than the usual 10 replaceable parameters (0%-9%) in batch file processing.

Parameters

None

Notes and Suggestions

- To see how SHIFT works, suppose a batch file, Money.bat, contains replaceable parameters, defined on the left in the following chart. When you enter the SHIFT command, each definition shifts one place. Instead of replacing %n, it replaces %n-1.

Definitions Before Shift	Definitions After Shift
%0 = money	%0 = pharm.dat
%1 = pharm.dat	%1 = autodeal.dat
%2 = autodeal.dat	%2 = hardware.dat
%3 = hardware.dat	%3 = grocery.dat
%4 = grocery.dat	%4 = dimestor.dat
%5 = dimestor.dat	%5 = theater.dat
%6 = theater.dat	%6 = cafe.dat
%7 = cafe.dat	%7 = photo.dat
%8 = photo.dat	%8 = beauty.dat
%9 = beauty.dat	%9 = undefined

As you can see, %9 is now open to be redefined. Money, on the other hand, no longer replaces any parameter. If you must refer to Money again, there are two ways to do it: Use SHIFT so that Money replaces %9, or use SET to set Money equal to a text parameter.

- You can execute as many shifts as needed. If you do, however, include ECHO %n in your file as needed to keep track of each replaceable parameter's current definition.

Examples

SHIFT

shifts all parameters that replace the parameters %0 through %9 down one parameter. Suppose that, before the shift, %1 equals Denfile.dat and %2 equals Myfile.dat. After the shift, %0 equals Denfile.dat and %1 equals Myfile.dat.

```
COPY con shiftcop.bat 
REM Stop execution at pause after all files are
copied 
:PAUSE 
PAUSE 
COPY A:\USER\%1 B:\USER\%1 
SHIFT 
GOTO PAUSE 
 
```

is a batch file that lets you copy as many files as you wish from A:\USER to B:\USER. It substitutes the first given file for %1 and copies it. Then it substitutes the next file, copies it, and so on. In this way, you can use SHIFT to gain access to more than 10 replaceable parameters. Even if you don't need that many parameters, however, you can use SHIFT to save typing time.

To copy the files Find.exe, Name1.asm, and Name2.asm from A:\USER to B:\USER, enter the name of the batch file, followed by the files to copy, as shown here:

```
shiftcop find.exe name1.asm name2.asm 
```

The batch file pauses before each copy, giving you the option to stop. After it copies all files, press at the PAUSE prompt (Strike a key when ready...).

SHIPTRAK

External

SHIPTRAK

Parks the heads of a hard disk at the innermost tracks in preparation for moving the drive unit.

Parameters

None

Notes and Suggestions

- Jarring your hard disk can cause its recording heads to bump against the highly polished surface of the recording media, destroying stored data. Such jarring can easily happen when you move your hard disk drive.

SHIPTRAK moves all your hard disk's recording heads onto the innermost tracks where information is not stored, and where such inadvertent bumping cannot destroy data.
- Always use SHIPTRAK before relocating your hard disk or anytime you think it might be bumped or jiggled.
- Using SHIPTRAK causes your computer to *lock up*. To use the computer again, turn off the power, wait at least 15 seconds, and then turn on the power again.
- Your hard disk is a precision instrument, built to extremely close tolerances. Always handle it very carefully, even after parking its heads with SHIPTRAK.

Example

```
SHIPTRAK [ENTER]
```

parks your hard disk heads at the innermost tracks.

SORT

External

SORT [/R] [/+*n*] [<*input pathname*]
[>*output pathname*]

SORT is a filter that reads input from the keyboard or a file, sorts the data, and writes it to the display or to a file.

Parameters

<*input
pathname* specifies the file to be sorted. If you omit this parameter, SORT sorts keyboard input.

>*output
pathname* specifies the file to receive the sorted information. If you omit this parameter, SORT sends the output to the display.

/R reverses the sort (sorts from Z to A). If you omit this parameter, the sort is from A to Z.

/+*n* begins the sort at Column *n*. If you omit this parameter, the sort begins at Column 1.

Notes and Suggestions

- Use SORT to alphabetize a file by a certain column. Suppose you have this information in a file, Mail.dat, on the current drive.

```
Jan King  2  8th St.   Lincoln    NE 68502
Sam Beck  4  6th St.   Rapid City SD 57001
Sal Cleo  7  9th St.   Ft. Worth  TX 76133
```

To sort the file alphabetically, according to last name, and then store the sorted output in another file, type:

```
SORT /+5 <mail.dat >sortmail.dat
ENTER
```

SORT sorts the contents of Mail.dat, starting at Column 5 (last name). It then outputs the sorted data into the file Sortmail.dat.

Examples

```
SORT /R <unsort.txt >sort.txt 
```

reads the file Unsort.txt, reverses the sort, and then writes the output to a file named Sort.txt.

```
DIR | SORT /+14 
```

pipes the output of the DIR command to the SORT filter, which sorts the directory listing, starting at Column 14 (file size). SORT then displays the output.

```
DIR | SORT /+14 | MORE 
```

is the same as the previous command, except that the MORE filter displays the directory one screen at a time.

SPOOLER

External

SPOOLER [/printer] [/P] [/S] [/C] [/G]

Lets you send commands to and get the status of the print spooler, assuming the SPOOLER.SYS device drive exists in your CONFIG.SYS file.

Parameters

<i>printer</i>	is the number of the printer you want to use. It can be either 1 or 2. If you omit the number, the system assumes 1. Notice that you must precede the number with a slash (/). You can install two spoolers if you have two printers connected.
/P	pauses the spooler. Using the /P switch is similar to toggling the off-line switch on the printer. Using /P a second time turns off the pause. The pause goes off automatically when the printer buffer is full to prevent the computer from locking up.
/S	interrupts the spooler's buffering so that your computer immediately stops printing data in the buffer. All printer data goes directly to the printer. This feature enables you to print data without waiting until the buffer is empty. Use /S again to restart the buffering.
/C	clears the spooler. Any data remaining in the buffer is not printed.
/G	returns the status of the spooler (installed, pause on/off, buffering on/off, size of buffer, percentage of buffer full).

Notes and Suggestions

- The spooler slows any programs that disable the spooler idle interrupt, such as programs written in BASIC.
- The spooler might slow the CPU clock slightly.

- See Appendix C for information on installing SPOOLER.SYS.

Examples

SPOOLER

starts print spooling for Printer 1.

SPOOLER /2 /P

pauses print spooling for Printer 2.

SPOOLER /2 /P

restarts print spooling for Printer 2.

SUBST

External

SUBST [*drive*] [*pathname*] [/D]

Substitutes a virtual drive name for a pathname.

Parameters

<i>drive</i>	is the virtual drive name. The highest available drive letter is the one specified in the CONFIG.SYS file with the LASTDRIVE command. If you have not used LASTDRIVE, the highest available letter is E.
<i>pathname</i>	is the path you want to refer to as <i>drive</i> .
/D	deletes the association between <i>drive</i> and <i>pathname</i> . Use /D when you no longer want to refer to <i>pathname</i> as <i>drive</i> .

Notes and Suggestions

- When MS-DOS sees a drive created with SUBST, it replaces the reference with the new pathname.
If you type
SUBST
MS-DOS displays the names of the current virtual drives.
- You cannot use SUBST over a network.

Examples:

```
SUBST Z: B:\USR\FRED\forms
```

creates a virtual drive Z for the pathname B:\USR\FRED\FORMS. Now, instead of typing the full pathname, you can get to this directory by typing Z:.

SYS (System)

External

SYS drive

Transfers the MS-DOS system files from the current disk to a formatted disk in the specified drive.

Parameters

drive is the drive containing the diskette to receive the files.

Notes and Suggestions

- SYS is normally used to update the system or to place the system on a formatted disk that contains no files. If the system files are already on the target disk, they must take up the same amount of space as the new system requires.
- The MS-DOS system files are hidden files that do not appear when you execute the DIR command.
- SYS does not transfer COMMAND.COM (the command processor). You must use the COPY command to transfer COMMAND.COM.
- You cannot use SYS over a network.
- Data disks and many application program disks do not contain the system. You can use the disk as is only if you have a system disk in Drive A. If you put the system files on the disk, however, you can use the disk alone.

Examples

SYS B:

transfers the system files from the current disk to the disk in Drive B.

TIME

Internal

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

Displays or sets the time. You can change the time from the keyboard or from a batch file. (Normally, MS-DOS displays a time prompt each time you start up your system. It does not, however, if you use an Autoexec.bat file. Therefore, you might want to include a TIME command in that file.)

Parameters

hh:mm:ss.cc specifies the time to set.

hh = 0-23 (hours)

mm = 0-59 (minutes)

ss = 0-59 (seconds)

cc = 0-99 (hundredths of a second)

If you include only part of the information (such as the hours) and press **ENTER**, the other fields are set at zero.

If you omit the time, TIME displays the current time and prompts you to enter the new time. Enter it in the 24-hour format, or press **ENTER** if you do not want to change the time displayed.

Notes and Suggestions

- When you change the time known to the system, you also change the time in any application program you use. This can be very handy.

Suppose you have a program that keeps track of customer calls according to the date and time received. For some reason, you get behind and cannot enter the information at the correct time. Simply enter it later, after “turning back the clock” to the necessary time.
- You can also use DATE and TIME to include the date and time on a printout. (See the DATE command.)

- TIME lets you keep track of the time, and can give you an idea how long it takes to run a particular program.
- TIME does not change the system CMOS time setting, which is displayed on startup.
- You can change the format for entering the time by using the COUNTRY command in your CONFIG.SYS file. See Appendix C, “Configuring Your System,” for more information.

Examples

```
TIME 14 
```

sets the time to 2:00 p.m.

```
TIME 2:32 
```

sets the time to 2:32 a.m.

TREE

Internal

TREE [*drive*][*/F*]

Displays all directories, subdirectories, and (optionally) all files on the disk in the specified drive.

Parameters

- drive* is the drive that contains the diskette you wish to examine. If you do not specify *drive*, the current drive is assumed.
- /F* causes the TREE command to display the names of all files in all levels of subdirectories.

Notes and Suggestions

- The TREE command lists the full pathname of each directory, along with the names of their subdirectories.
- Using a redirection symbol, you can reroute TREE output to another device, such as disk or printer.

Examples

```
TREE B: 
```

displays all the directories at all levels contained on the diskette in Drive B.

```
TREE A:/F>PRN 
```

lists all subdirectories and filenames on the diskette in Drive A to the printer.

```
TREE A:/F>Adir.fil 
```

sends a list of all the subdirectories and files on Drive A to the disk file Adir.fil on the current directory of Drive A.

TYPE

Internal

TYPE *pathname*

Displays the contents of the specified file, without modifying the file. If the file contains tabs, TYPE uses eight spaces to represent each tab.

Parameters

pathname is the name of the file to list.

Notes and Suggestions

- A display of a binary file sends control characters (such as end-of-file characters, bells, form feeds, and escape sequences) to your computer.
- Use DIR to display the name of a file, and use TYPE to display the file's contents. This is a good way to see if the file requires editing. If it does, you can use EDLIN to do the editing. (See Part 3 for information on editing.)

Examples

```
TYPE \USER\taxfile.dat 
```

displays the file Taxfile.dat that is in the USER directory in the current drive.

```
TYPE B:carfile 
```

displays the file Carfile that is in the home directory of Drive B.

```
TYPE B:\BUDGET\clothes 
```

displays the file Clothes that is in B:\BUDGET.

VER (Version)

Internal

VER

Displays the number of the MS-DOS version that you are using.

Parameters

None

Notes and Suggestions

- If you have a question or comment about your system, you need to know the MS-DOS version number when you contact your Customer Services Department.

Example

VER

displays the version number.

VERIFY

Internal

VERIFY [ON|OFF]

Turns the verify switch on or off, or displays the current setting of VERIFY. VERIFY has the same purpose as the /V switch in the COPY command. When it is on, it tells MS-DOS to verify that your files are intact (that they contain no bad sectors, for example).

Parameters

ON|OFF determines the setting of VERIFY. The setting remains in effect until you change a program (by using a SET VERIFY system call), or until you use VERIFY command again. When VERIFY is on, it verifies all writes to disk.

If you omit ON and OFF, MS-DOS returns the current setting of VERIFY.

Notes and Suggestions

- By keeping VERIFY on, you always know if your files are intact. Note, however, that this slows operation.
- You receive an error message only if MS-DOS is unable to write your data to disk successfully.

Examples

```
VERIFY ON 
```

tells MS-DOS to verify all writes to disk.

```
VERIFY OFF 
```

tells MS-DOS to stop verifying writes to disk.

VOL (Volume)

Internal

VOL [*drive*]

Displays the volume label of the disk in the specified drive. (You can assign a label by using the /V parameter in the FORMAT command.) If the disk does not have a volume label, VOL displays:

```
Volume in drive n has no label
```

Parameters

drive indicates the disk for which you wish to display the label. If you omit the drive, MS-DOS displays the volume label of the current disk.

Notes and Suggestions

- Enter the VOL command whenever you forget a volume label.

Example

```
VOL B: 
```

MS-DOS displays a message similar to this:

```
Volume in drive B is DISKONE
```

```
VOL 
```

displays the volume label of the current disk.

XCOPY

External

XCOPY *source pathname* [*target pathname*] [/A]
[/D:*date*] [/E] [/M] [/P] [/S] [/V] [/W]

Copies either a file or a directory and (optionally) the directory's subdirectories from a disk to another disk of any type. Unlike DISKCOPY, XCOPY does not require that the source and target disks have the same format.

Parameters

<i>source pathname</i>	is the name of the file or directory you want to copy. If you omit the directory part of <i>source pathname</i> , XCOPY copies the file from the current directory. If you omit the filename part of <i>source pathname</i> , XCOPY uses the wildcard *.* and copies all files in the directory.
<i>target pathname</i>	is the name to assign the copy. If you omit the directory part of <i>target pathname</i> , XCOPY copies the file to the current directory.
/A	copies only source files that have their archive bit set. It does not change the archive bit of the source file. (See ATTRIB for information on how to set the archive bit.)
/D: <i>date</i>	copies only those source files modified on or after <i>date</i> . The date format varies, depending on the country code that you are using.
/E	copies all subdirectories of the specified directory, including empty subdirectories. You must use /S if you use /E.
/M	copies only source files that have their archive bit set, and turns off the archive bit in the source files. (See ATTRIB for information on how to set the archive bit.)
/P	prompts you so that you can confirm whether you want to create each target file.

/S copies the specified directory and its subdirectories (as long as the subdirectories are not empty). When used with **/E**, **/S** copies the empty subdirectories, as well. If you omit **/S**, **XCOPY** works within one directory.

/V causes **XCOPY** to verify each target file to be sure it is identical to the source file.

/W causes **XCOPY** to wait before it starts copying the files. **XCOPY** displays the message:

Press any key when ready to start
copying files

Press any key to continue, or press **CTRL C** to
cancel the copy.

Notes and Suggestions

- The **XCOPY** command might prompt you to specify whether the target is a file or a directory. If you don't want to receive this prompt, type:

COPY /B XCOPY.EXE MCOPY.EXE **ENTER**

The preceding command creates a new command called **MCOPY.EXE**. Now, you can use the **MCOPY** command the same way you use **XCOPY**, except that you won't receive the prompt.

- **MCOPY** uses the following rules for copying files:
 - If the source is a directory, the target is a directory.
 - If the source includes multiple files, the target is a directory. For example, if the source is **A:**, and Drive **A** contains more than one file, the target is a directory.

- If you append a backslash (\) to a target name, that target is a directory. For example:

```
XCOPY payroll A:\WORKERS\ [ENTER]
```

creates the directory A:\WORKERS if the directory doesn't already exist, and copies the file Payroll to it.

Examples

```
XCOPY A: B: /S /E [ENTER]
```

copies all the files and directories (including empty subdirectories) on Drive A to Drive B.

```
XCOPY jobstat B:\STATUS\ /V [ENTER]
```

copies the file Jobstat from the current directory to the STATUS directory of Drive B, and verifies that the source and target files are identical.

```
XCOPY A: B: /A [ENTER]
```

copies all read-only files in the current directory of Drive A to Drive B.

```
XCOPY A: B:\MODFILES\ /D:11/15/86 [ENTER]
```

copies from the current directory of Drive A all files modified on or after November 15, 1986, and places them in the \MODFILES directory on Drive B.

COMMAND EDITING

MS-DOS has several special editing features to help you change commands. Learning to use these features can save you time.

The Template

As mentioned in Chapter 1, you can edit commands in MS-DOS through a special storage area called the template. When you press **ENTER** after typing a command, MS-DOS sends the command to the command processor (COMMAND.COM) for execution. It also sends the command to the template. The template stores only one command at a time—the last command entered. You can then recall the command from the template for editing or re-execution.

Function Keys

The following table illustrates the function keys you can use to help you edit efficiently.

Function	Key(s)	Description
Copy <i>char</i>	F1 or →	copies a character from the template to the command line and displays it.
Delete <i>char</i>	DEL	deletes a character from the template. The character is skipped and not copied to the command line.
Copy to <i>char</i>	F2 <i>char</i>	copies all characters up to the specified character and displays them.
Delete to <i>char</i>	F4 <i>char</i>	deletes all characters up to the specified character from the template. They are skipped and not copied to the command line.

Function	Key(s)	Description
Copy all	F3	copies all remaining characters and displays the entire command line.
Insert	INS	enters the insert mode. Press F3 to exit.
Replace template	F5	makes the new line the new template, but does not execute the command (accepts the line for more editing).
Void line	ESC	exits the current input. Leaves the template unchanged. Type and enter a new line, or press ENTER to display the system prompt.
Enter line	ENTER	makes the new line the new template and sends it to the requesting program.
End-of-file	F6 or CTRL Z	puts an end-of-file character in the template.

Sample Session

The rest of this chapter is a sample session, provided to quickly acquaint you with all the functions available for editing command lines.

Display and Change

Assume you have two files, Prog.com and Prog.asm, in the ROOT directory of your MS-DOS disk. If you type the following command:

```
DIR prog.com ENTER
```

MS-DOS displays information about the file Prog.com. At the same time, it saves the command line (DIR Prog.com) in the template.

Press **[F3]** to display the command line. (The underscore is used here to represent the blinking cursor.) The screen displays:

```
DIR prog.com _
```

Execute the command again by pressing **[ENTER]**

To display information about the file Prog.asm, edit the command line DIR Prog.com. Type:

```
[F2] C
```

MS-DOS displays all characters of the command line up to but not including the letter C, as shown.

```
DIR prog._
```

Now type:

```
asm_
```

The letters asm replace the letters com, resulting in:

```
DIR prog.asm_
```

This new command line is now in the template. To execute it, press **[ENTER]**.

Overtyping and Insert

Replace DIR with the TYPE command by typing:

```
TYPE_
```

The characters TYPE automatically replace the characters DIR and the space that followed.

Now press **[INS]**, the space bar, and **[F3]**

Pressing **[INS]** and the space bar inserts a space. **[F3]** copies the rest of the template. The result is:

```
TYPE prog.asm_
```

To execute the new command, press **[ENTER]**.

Saving in the Template

Suppose you misspell `TYPE` as `BYTE`, but have not entered the command. (Set up this situation by typing `BYTE` and pressing `[F3]` again.) Your screen displays:

```
BYTE prog.asm__
```

To avoid retyping the entire command, press `[F5]`. The symbol `@` appears at the end of the line. This indicates that MS-DOS has put the new line in the template, although it has not sent it to be executed.

Now edit the line `BYTE Prog.asm` so it becomes `TYPE Prog.asm`. To do so, type:

```
T↵P[F3]
```

The `↵` copies a single character `Y` from the template to the command line. The resulting command line is:

```
TYPE prog.asm__
```

Deleting

You can make the same change another way. To edit the line `BYTE Prog.asm`, press `[F5]` again. Then type:

```
[DEL] [DEL] ↵ [INS]YP[F3]
```

Pressing `[DEL]` twice deletes the first two characters (`B` and `Y`). The `↵` key copies the third letter, `T`. `[INS]` enters the insert mode so that you can insert the letters `Y` and `P`. `[F3]` copies the rest of the template.

`[DEL]` does not affect the command line. It affects the template by deleting the first character. `[F4]` deletes characters in the template, up to but not including a given character.

FILE EDITING

The techniques for editing command lines also apply to editing files. You can use the same function keys (listed in Chapter 7). In addition, there are several other special features available for creating, changing, and manipulating blocks of text.

Files created with EDLIN are divided into lines with a maximum length of 253 characters. During editing, EDLIN generates and displays a number for each line. If you insert or delete lines, it automatically renumbers any following lines to maintain consecutive numbering. The line numbers are only to aid in editing; they are not actually in the file.

Creating a File

To use EDLIN to create a file, you simply type:

```
EDLIN pathname 
```

where *pathname* refers to a file that you wish to create. In this case, create a sample file for editing by typing:

```
EDLIN sampfile 
```

Note: If you wish to have the file reside in a directory other than the home directory, give the *pathname* of desired directory. The following command, for example, tells EDLIN to create the file in the home directory of Drive B:

```
EDLIN B:sampfile 
```

In either case, EDLIN displays the message and prompt:

```
New file
*__
```

to indicate it has created the sample file. You are now in the EDLIN command mode, and can enter any of the commands outlined in Chapter 10.

Inserting Text

1. Type I to begin inserting text. The screen displays:

```
New file
*I
1:*__
```

2. On Line 1, type:

```
This is a sample file 
```

EDLIN displays the number for the next line:

```
2:*__
```

3. Type:

```
The editing keys are easy to use 
```

4. When EDLIN displays the next line number, type . This exits the insert mode and returns the EDLIN prompt (*).

5. Type I to tell EDLIN to display Line 1. The screen looks like this:

```
*1
1:*This is a sample file
1:*__
```

You can enter a maximum of 254 characters into each line. To end one line and start another, press . EDLIN places the current line in the template and displays the next line number.

Saving Your File

Now, save your newly created file by pressing to exit the insert mode and then typing E to save the file and return to the MS-DOS command prompt. When you save a newly created file, EDLIN gives the file the filename and extension (if any) you specified when you created it.

Editing an Existing File

You can now recall your newly created file, or any EDLIN file. If you specify a file that already exists when you use the EDLIN command, EDLIN loads it into memory and then displays the message:

```
End of input file
*__
```

Note: If the file is large, EDLIN loads text until memory is 75% full. When it displays the * prompt, you can edit those lines. To edit more of the file, first free some memory by writing the edited lines to disk. (See the Write Lines command in this chapter.) Then load more of the file for editing. (See the Append Lines command.)

When you have completed editing, save both the edited version of the file and the original file by typing E . However, the original file is now saved with the extension .bak and the edited version is saved with the original name and extension.

Note: You cannot further edit a file that has a .bak extension. To overcome this system restriction, rename the file, giving it another extension. (See RENAME in Part 2.)

Using the Special EDLIN Editing Keys

You can now use the special edit keys to change the file you created. Recall your EDLIN file by typing:

```
EDLIN sampfile 
```

EDLIN displays:

```
End of input file
*__
```

To display Line 1 of the file, type:

```
1 
```

You are now ready to begin editing.

Example 1: Copy *Char* (**F1**)

Using the Copy *Char* function, **F1**, copy characters from the template to the new line.

1. Press **F1** once. The screen looks like this:

```
1:*This is a sample file
1:*T__
```

2. Press **F1** 3 more times. The screen looks like this:

```
1:*This is a sample file
1:*This__
```

Each time you press **F1**, EDLIN copies another character from the template to the new line.

3. To continue to the next example, press **CTRL C**. This exits the insert mode, voids any changes you have made to the line, and returns the EDLIN prompt (*).

Example 2: Copy to *Char* (**F2**)

Using the Copy to *char* function, **F2**, copy all characters from the template, up to—but not including—the specified character.

1. At the EDLIN prompt, type **2 ENTER**. The screen looks like this:

```
2:*The editing keys are easy to use
2:*__
```

2. Type **F2 a**. The screen looks like this:

```
2:* The editing keys are easy to use
2:* The editing keys __
```

3. Now copy the rest of the template, using the Copy All function, **F3**. The screen looks like this:

```
2:*The editing keys are easy to use
2:*The editing keys are easy to use
```

4. Press **CTRL C** to continue to the next example.

Note: If the template does not contain the specified character, nothing is copied.

Example 3: Copy All (**F3**)

Using the Copy All function, (**F3**), copy the entire contents of the template to the new line.

1. Type (**1**) (**ENTER**) at the EDLIN prompt (*). The screen looks like this:

```
1:*This is a sample file
1:*__
```

2. Press the Copy All editing key, (**F3**). The screen looks like this:

```
1:*This is a sample file
1:*This is a sample file__
```

3. Press (**CTRL**) (**C**) to continue to the next example.

Note: The Copy All function copies all characters in the template to the new line. However, if you changed part of the template, the Copy All function copies only the remaining characters—those that have not been edited.

Example 4: Delete *Char* (**DEL**)

Using the Delete *Char* function, delete some characters from the template.

1. At the EDLIN prompt, type (**2**) (**ENTER**). The screen looks like this:

```
2:*The editing keys are easy to use
2:*__
```

2. Press (**DEL**) 4 times to delete the first 4 characters from the template. Then press the Copy All function key, (**F3**). The screen looks like this:

```
2:*The editing keys are easy to use
2:*editing keys are easy to use __
```

3. Press (**CTRL**) (**C**) to void any changes to Line 2 and continue to the next example.

Example 5: Delete to *Char* (**F4**)

Using the Delete to *Char* function, **F4**, delete a number of characters up to—but not including—the specified character.

1. At the EDLIN prompt, type **1** **ENTER**. The screen looks like this:

```
1:*This is a sample file
1:*__
```

2. Type **F4** **a**. The cursor remains in the same position even though the characters before the specified character are deleted. To see this, press the Copy All key, **F3**. The screen looks like this:

```
1:*This is a sample file
1:*a sample file__
```

3. Press **CTRL** **C** to void any changes and continue to the next example.

Example 6: Void Line (**ESC**)

In this example, first make a change to Line 1 and then, using the Void Line function, **ESC**, cancel that change. This leaves the template unchanged and lets you continue making changes to the same line.

1. At the EDLIN prompt, type **1** **ENTER**. The screen looks like this:

```
1:*This is a sample file
1:*__
```

2. Type:

```
Names and addresses ESC
```

The screen looks like this:

```
1:*This is a sample file
1:*Names and addresses\
__
```

When you press **ESC**, a backslash (\) appears right after any change you may have made and the cursor appears again on another line.

3. Press the Copy All key, **[F3]**. The screen looks like this:

```
1:*This is a sample file
1:*Names and addresses\
   This is a sample file__
```

4. Press **[CTRL]** **[C]** to continue to the next example.

Example 7: Insert (**[INS]**)

The **[INS]** key acts as a switch to turn the insert mode on and off. Use this key to insert a word in Line 1 of the sample file.

1. At the EDLIN prompt, type **[1]** **[ENTER]**. The screen looks like this:

```
1:*This is a sample file
1:*__
```

2. Press the Copy to *char* key, **[F2]** and press **[1]**. The screen looks like this:

```
1:*This is a sample file
1:*This is a sample __
```

3. Now press **[INS]** to turn on the insert mode. Type *edit* followed by a single space. Then press **[INS]** again to turn off the insert mode. The screen looks like this:

```
1:*This is a sample file
1:*This is a sample edit __
```

4. To copy the rest of the characters in the template, press the Copy All key, **[F3]**. The screen looks like this:

```
1:*This is a sample file
1:*This is a sample edit file__
```

5. Press **[CTRL]** **[C]** to continue to the next example.

Example 8: Replace Template (**F5**)

Using the Replace Template function, **F5**, replace the template with whatever characters are displayed in the new line.

1. At the EDLIN prompt, type **2 ENTER**. The screen looks like this:

```
2:*The editing keys are easy to use
2:*__
```

2. Type:

```
Replacing the template
```

Then press **F5**. Notice that the @ symbol appears after the last character and the cursor moves to the next line:

```
2:*The editing keys are easy to use
2:*Replacing the template@
__
```

After you press **F5**, all the characters displayed in the new line replace those in the template. This is similar to pressing **ENTER**. However, there is one important difference. With **F5**, the displayed characters go only to the template for further editing; they do not go to the requesting program.

3. Press the Copy All function key, **F3**. The screen looks like this:

```
2:*The editing keys are easy to use
2:*Replacing the template@
Replacing the template __
```

As you can see, the new sentence is now the template.

4. Press **CTRL C** to void any changes and continue to the next example.

Note: Pressing **ENTER** immediately after pressing **F5** empties the template.

Example 9: Enter Line (ENTER)

Whenever you press **ENTER**, the displayed characters become the template and are sent to the requesting program.

1. At the EDLIN prompt, type **2** **ENTER**. The screen looks like this:

```
2:*The editing keys are easy to use
2:*__
```

2. Press the Copy to *char* key, **F2**, and then press **a**. The screen shows:

```
2:*The editing keys are easy to use
2:*The editing keys __
```

3. Then type:

```
simplify editing tasks ENTER
```

The screen looks like this:

```
2:*The editing keys are easy to use
2:*The editing keys simplify editing tasks
*__
```

Notice that after you press **ENTER** the EDLIN prompt returns to the screen.

4. Type **2** **ENTER**. The screen looks like this:

```
2:*The editing keys simplify editing tasks
2:__
```

As you can see, Line 2 now contains the new sentence you entered.

INTRODUCTION TO EDLIN REFERENCE

In addition to the editing keys, EDLIN supports many powerful commands that let you manipulate an entire file or several lines at a time.

Summary of EDLIN Commands

The following table summarizes the EDLIN commands:

Command	Purpose
<i>line</i>	Edits the specified lines
A	Appends lines
C	Copies lines
D	Deletes lines
E	Ends editing
I	Inserts lines
L	Lists text
M	Moves lines
P	Pages text
Q	Quits editing
R	Replaces lines
S	Searches text
T	Transfers lines
W	Writes lines

Command Format And Rules

Most EDLIN commands follow a certain format and conform to certain rules. Following is a brief review:

- Except for Edit Line, End Edit, and Quit, EDLIN commands are preceded and/or followed by parameters. The function of parameters is discussed later.
- You can enter EDLIN commands in uppercase or lowercase, or a combination of upper- and lowercase characters.
- Except for the Edit Line command, EDLIN commands are a single letter.

- You can indicate line numbers relative to the current line. A minus sign (-) indicates lines that precede the current line. A plus sign (+) indicates lines that follow it. For example: -10, +10L **ENTER** lists the 10 lines immediately preceding the current line, the current line, and the 10 lines immediately following the current line.
- Usually, you can enter multiple commands in the same line without using any special delimiting characters. The exceptions are:
 - when you use the Edit Line command for a single line.
 - when you use the Search and Replace command.

In the first case, you must use a semicolon to separate the commands. For example, the multiple command: 15; -5, + 5L **ENTER** edits Line 15 and then displays Lines 10-20.

In the second case, press **CTRL** **Z** (instead of **ENTER**) after the string you want to replace. For instance, the multiple command:

```
SThis string CTRL Z -5, + 5L ENTER
```

searches for the string, This string, and then displays the five lines that immediately precede the matched string and the five lines that immediately follow it. If the search fails, then EDLIN displays the numbers of the relative lines.

- You can enter EDLIN commands with or without a space between the line number and the command. For example, the delete command 6D is the same as 6 D.
- You can insert a control character (such as **CTRL** **C**) into text by using the special quote character, **CTRL** **V**, before the control character while in the insert mode. **CTRL** **V** tells MS-DOS to recognize the next **uppercase** letter typed as a control character. You can even use a control character in a Search or Replace command, by using **CTRL** **V**. For example:

```
SCTRL V Z
```

finds the next occurrence of **CTRL** **Z** in a file.

It is possible to insert **CTRL** **V** into text by typing **CTRL** **V** V.

- Delimiters (spaces or commas) are required only between adjacent line numbers. You can also use them, however, to separate commands and parameters for better readability.
- Commands become effective after you press `ENTER`.
- For commands that produce a large amount of display output, press `CTRL S`. This halts the display so you can read it. Press the space bar to continue display output.
- You can stop EDLIN commands by pressing `CTRL C`.

Command Parameters

Most EDLIN commands require you to specify a line, a string, or a range of lines to be affected by the command. These are the *command parameters*. Command parameters might vary according to the EDLIN command, but, generally, they are as follows:

line (line number)

This parameter indicates that you must specify a line number for the command to affect. You can specify a line in three ways:

- By using the actual line number. This number must be less than 65534. If you specify a number larger than the number of existing lines, EDLIN adds a line to the file.
- By using a period (.) as shorthand notation to indicate the current line. The current line is the last line edited, not necessarily the last line displayed. It is always marked on your screen by an asterisk between the line number and the first character.
- By using a pound sign (#) as shorthand notation to indicate the last line of the program.

Separate line numbers (and line number notations) from commands, parameters, and other line numbers with a comma or a space.

***string* (a group of characters)**

This parameter represents a portion of text EDLIN is to find, replace, delete, or use to replace other text. Use this parameter only with the Search and Replace commands. End each string with `CTRL Z`. Do not include spaces between strings or between a string and the command letter (unless you want those spaces to be part of the string).

EDLIN COMMAND REFERENCE

Append Lines (A)

[*number*]A

Adds the specified number of lines from disk to the file being edited in memory. EDLIN appends the lines to the end of the file.

This command is meaningful only if the file you are editing is too large to fit into memory. Before using this command, you must write the portion of the file that was loaded into memory (and which has been edited) to disk. Refer to the Write Lines command.

When the Append command reads the last line of the file into memory, EDLIN displays the message:

End of input file

If you omit the number of lines, EDLIN appends lines until available memory is 75% full. However, if memory is already 75% full, EDLIN does nothing.

Example:

100A

appends 100 lines from the disk to the file in memory.

Copy Lines (C)

[*line1*][,*line2*],*line3*[, *count*]C

Copies all lines ranging from *line1* to *line2*, and places the duplicate lines immediately ahead of *line3*. Using the *count* parameter, you can copy the lines as many times as you want.

If you omit *line1* or *line2*, EDLIN copies the current line. If you omit the count, EDLIN copies the line(s) only once.

Note: If you omit the *line1* parameter, your command must include a comma immediately preceding the *line2* parameter.

EDLIN automatically renumbers the file, and makes the first line copied the current line. For example:

```
3,9,12C 
```

copies Lines 3-9, and places them immediately ahead of Line 12. The new Line 12 becomes the current line.

Note: Line numbers must not overlap. If they do, EDLIN displays an error message. Also, you must not use the plus and minus signs.

Examples:

Assume that the following file exists and is ready for editing. If you want to create the file, see the Insert command.

```
1: This sample file
2: shows what happens
3: when you use
4: the Copy command
5: to copy text in your file.
```

To copy the above block of text, type the following command at the EDLIN prompt:

```
1,5,6C 
```

Almost instantly, the prompt reappears to indicate that EDLIN is finished executing the command. If you wish to see the result, type `[L][ENTER]`. The screen shows:

```
1: This sample file
2: shows what happens
3: when you use
4: the Copy command
5: to copy text in your file.
6:* This sample file
7: shows what happens
8: when you use
9: the Copy command
10: to copy text in your file.
```

Notice that the first line copied becomes the current line.

If you want to insert lines between other lines, use *line3* to specify the line before which you want the text copied. For example, assume that you want to insert Lines 5-8 immediately preceding Line 3 in the following file:

```
1: This sample file
2: shows what happens
3: when you use the Copy command
4: to copy text in your file.
5: If you so choose,
6: you can also insert
7: a group of lines within
8: other parts of the file.
9: The Copy command
10: is versatile.
```

The command 5,8,3C ENTER results in the following file:

```
1: This sample file
2: shows what happens
3:*If you so choose,
4: you can also insert
5: a group of lines within
6: other parts of the file.
7: when you use the Copy command
8: to copy text in your file.
9: If you so choose,
10: you can also insert
11: a group of lines within
12: other parts of the file.
13: The Copy command
14: is versatile.
```

Delete Lines (D)

[*line1*][,*line2*]D

Deletes *line1* or all lines in the range *line1* to *line2* from a file.

EDLIN automatically renumbers the lines to maintain consecutive numbering in the file.

Parameters

You can omit the *line1* parameter, as in:

*,line2*D

If you do, EDLIN starts deleting at the current line and ends at *line2*. Your command line must include the comma before *line2* to indicate you are omitting *line1*.

You can omit the *line2* parameter, as in:

*line1*D or
line1,D

If you do, EDLIN deletes only *line1*.

If you omit *line1* and *line2*, as in:

D

EDLIN deletes only the current line.

Examples:

Assume that the following file exists and is ready to be edited.
Line 30 is the current line:

```
1: This sample file
2: shows how
3: the Delete command functions.
4: All you need to do
5: in most cases
.
.
.
26: is specify
27: a number of lines
28: to delete;
29: deleting often helps
30:*clean up your files.
```

Type:

5,25D

The result is:

```
1: This sample file
2: shows how
3: the Delete command functions.
4: All you need to do
5:*is specify
6: a number of lines
7: to delete;
8: deleting often helps
9: clean up your files.
```


To delete Line 4 in the above file, type:

4D

The result is:

```
1: This sample file
2: shows how
3: the Delete command functions.
4:*is specify
5: a number of lines
6: to delete;
7: deleting often helps
8: clean up your files.
```

To delete the current line and the following two lines, type:

,6D

The result is:

```
1: This sample file
2: shows how
3: the Delete command functions.
4:*deleting often helps
5: clean up your files.
```

To delete only the current line, type:

D

The result is:

```
1: This sample file
2: shows how
3: the Delete command functions.
4:*clean up your files.
```

Edit Line (*line*)

[*line*]

Lets you load the specified line for editing.

EDLIN displays the specified line, drops down one line and displays the line's number (without text) to indicate the line is ready for editing. You can then use the editing keys, control keys, and the EDLIN commands.

Notes

- If you don't specify the line (you only press `[ENTER]`), EDLIN loads the line that immediately follows the current line.
- If you don't need to change the line, and the cursor is at the beginning or end of the line, press `[ENTER]` to accept the line as is.
- If you press `[ENTER]` while the cursor is in the middle of a line, EDLIN erases the rest of the line.
- If you wish to cancel any changes made to a line, press `[ESC]`.

Example

Assume the following file exists, and you wish to edit Line 4:

```
1: This is a sample file.  
2: used to show  
3: the editing of line  
4: *four.
```

Type `4[ENTER]` at the EDLIN prompt. The screen looks like this:

```
*4  
4: *four  
4: ____
```

Press `[INS]`, type the word `number`, followed by a single space, and press `[INS]` again. Then, to copy the rest of the line, press the Copy All editing key, `[F3]`. The screen shows:

```
*4
4:*four
4:*number four__
```

At this point, you can:

- Save the changed line by pressing `[ENTER]`.
- Type more text at the end of the line. (The insert mode is in effect whenever the cursor is at the end of the line.)
- Press `[F5]` (Replace Template) to further edit the line.
- Press `[ESC]` to cancel the changes made to the line.

End Edit (E)

Ends the EDLIN program and saves the edited file.

When EDLIN saves the edited file, it uses the drive specification, filename, and extension you specified when you started EDLIN. (If you want, you can use the MS-DOS COPY command to transfer the file to another drive.)

If you edit an existing file (rather than a newly created one), EDLIN also saves the original file (unedited file). It gives the original file the extension .bak (for backup).

When you enter the End Edit command, EDLIN appends a CTRL
Z character to serve as the end-of-file mark.

After you enter the End Command, control returns to MS-DOS. The screen displays the system prompt (such as A>).

Warning: Be sure the disk contains enough free space to save the entire file. If it does not, EDLIN saves only part of the file; the rest is lost. If this occurs, the portion saved has the file extension .\$\$\$.

Insert (I)

[line]I

Lets yo insert text in a newly created file, starting at Line 1, or in an existing file, immediately preceding the line you specify.

The next line number appears automatically each time you press **ENTER**.

You **must** use the I command to begin writing text in a new file.

Notes

- If you use a period (.) to specify the line—or if you omit the line parameter—EDLIN uses the current line.
- If you use a pound sign (#) to specify the line, or if the line you specify is any number larger than the number of the last line—EDLIN appends the lines to the end of the file. In this case, the last line inserted becomes the current line.

Examples

Assume that the following file exists and is ready for editing:

```
1: This is a sample file
2: to show what happens
3: when you use the Insert command
4: in your files
```

To insert text before a line other than the current line, type the line number and I. Then, press **ENTER**. For example, type:

```
3 I ENTER
```

The result is:

```
3: *__
```

Now, type the new text for Line 3:

```
3: to the line numbers
```

To continue text insertion, press `ENTER`. EDLIN displays the next line number. Type:

4: (how EDLIN rennumbers them) `ENTER`

Then, return to the EDLIN prompt by pressing `CTRL C`. Type `L` `ENTER` to display the result:

```
1: This is a sample file
2: to show what happens
3: to the line numbers
4: (how EDLIN rennumbers them)
5: *when using the Insert command
6: in your files
```

To insert a line immediately before the current line, type `I` `ENTER` at the EDLIN prompt. The screen shows:

```
5: *__
```

Now type:

every time you insert new lines `ENTER`

When EDLIN displays the number for the next line, press `CTRL C` to return to the EDLIN prompt (*). Type `L` `ENTER` to list the file. The screen looks like this:

```
1: This is a sample file
2: to show what happens
3: to the line numbers
4: (how EDLIN rennumbers them)
5: *every time you insert new lines
6: when you use the Insert command
7: in your files
```

To add lines to the end of the file, type:

8I `ENTER`

The screen shows:

```
8: *__
```

Enter the following new lines:

```
8: You can use
9: the Insert command
10: to add new lines
11: to the end of your file
```

Press **CTRL C** when EDLIN displays the number for Line 12. Then, at the EDLIN prompt, type **L ENTER** to list the file. The screen looks like this:

```
1: This is a sample file
2: to show what happens
3: to the line numbers
4: (how EDLIN rennumbers them)
5: every time you insert new lines
6: when you use the Insert command
7: in your files
8: You can use
9: the Insert command
10: to add new lines
11: to the end of your file
```

List (L)

[*line1*][,*line2*]L

Displays all lines in the range *line1* to *line2*.

Notes

- You can omit *line1*, as in:

*,line2*L

If you do, the display begins before the current line and ends with *line2*. You must use the comma to indicate that you are omitting *line1*.

- You can omit *line2*, as in:

*line1*L

If you do, EDLIN displays 23 lines, starting with *line1*.

- If you omit both parameters, EDLIN displays 23 lines—the 11 lines immediately preceding the current line, the current line, and the 11 lines immediately following the current line. If fewer than 11 lines precede the current line, EDLIN displays more lines that follow the current line, to make a total of 23 lines.
- Even if a line is longer than the width of the screen (so that the line requires more than one display line), EDLIN counts it as a single line.
- If you can list more lines than the screen can display at one time, the excess lines scroll off the top of the screen. Press to pause the scroll. Press again to continue.

Examples

Assume that the following file exists and is ready for editing:

```
1: This is a sample file
2: used to show the List command.
3: See what happens when you use
4: List (L) with different parameters
5: or without any
.
.
.
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, type:

```
line1,line2L 
```

For example, type:

```
2,5L 
```

to produce the following display:

```
2: used to show the List command.
3: See what happens when you use
4: List (L) with different parameters
5: or without any
```

To list a range of lines beginning with the current line, type:

```
.,lineL 
```

For example, type:

```
.,26L 
```

to produce this display:

```
15:*The current line contains an asterisk.  
.  
.  
.  
26: to edit text
```

To list a range of 23 lines centered around the current line, you need only type L . The screen shows:

```
4: List (L) with different parameters  
5: or without any  
.  
.  
.  
.  
.  
15:*The current line contains an asterisk.  
.  
.  
.  
26: to edit text.
```

Move Lines (M)

[*line1*][,*line2*],*line3*M

Moves all lines in the range *line1* to *line2* to a position immediately preceding *line3*. *line1* becomes the current line.

Use the Move command to move a block of lines from one place in the file to another.

Notes

- If you omit the *line1* parameter or the *line2* parameter, EDLIN uses the current line.
- EDLIN rennumbers the lines according to the direction of the move. For example:

```
, + 25, 100M 
```

moves the text from the current line + 25 to Line 100. If the line numbers overlap, EDLIN displays the message:

```
Entry error
```

Example

To move Lines 20-30 to Line 100, type:

```
20,30,100M 
```

Page (P)

[*line1*][,*line2*]P

Pages through a file 23 lines at a time, or lists the specified block of lines.

Notes

- You can omit the *line1* parameter, as in ,*line2*P .
- If you do, EDLIN uses the current line number + 1, unless the current line is Line 1. You must include the comma to indicate that you are omitting the *line1* parameter.
- You may omit the *line2* parameter, as in ,*line1* P . If you do, EDLIN lists 23 lines.
- The last line displayed becomes the current line. This command differs from the List command in that the Page command changes the current line to the last line displayed.

Example

To display Lines 10-15, type:

10,15P

To display Lines 20-42, type:

20P

EDLIN displays the specified line (Line 20) and the next 22 lines. Line 42 becomes the current line.

Then, to display Lines 43-65, type:

P

EDLIN displays the current line + 1 (Line 43) and the next 22 lines. Line 65 becomes the current line.

Quit (Q)

Quits the editing session without saving any changes you might have made to the file.

EDLIN asks you if you want to save the changes. Press ☐ Y if you want to quit the editing session without saving any changes. Press ☐ N to continue editing.

Example

```
*Q ☐ ENTER
Abort edit (Y/N)___
```

Replace String (R)

[*line1*][,*line2*][?]R[*string1*] [CTRL] [Z] [*string2*]

Replaces all occurrences of *string1* with *string2*. The replacement is limited to the lines between *line1* and *line2*.

Notice that you must terminate *string1* by pressing [CTRL] [Z], and begin *string2* immediately following the [CTRL] [Z] character. Terminate *string2* by pressing [CTRL] [Z] or [ENTER].

Each time Replace finds *string1*, it replaces *string1* with *string2*. It displays the line affected. If a line contains more than one replacement, Replace displays it once for each occurrence.

When all occurrences of *string1* in the specified range are replaced with *string2*, the Replace command terminates, and the EDLIN prompt (*) reappears.

The question mark (?) tells Replace to display □.K.? each time it modifies a line. Press [Y] to accept the change. Press [N] to reject it. In either case, the search continues, starting at the next occurrence.

Notes:

- If you omit *line1*, Replace starts the search at the line immediately following the current line; it stops at *line2*.
- If you omit *line2*, Replace continues the search to the end of the file.
- If you omit both *line1* and *line2*, searching begins at the line that immediately follows the current line, and ends at the end of the file.
- If you omit *string2*, but you include *string1*, EDLIN **deletes** all occurrences of *string1* in the specified line range.
- If you omit *string1* and *string2*, EDLIN uses the most recent string specified with a Search (or Replace) command as *string1*, and the most recent string specified in a Replace command as *string2*.

Examples:

Assume that the following file exists and is ready for editing:

```
1: This sample file
2: shows how the Replace command works and how
3: when you use Replace
4: certain words and phrases
5: may be exchanged for
6: other words and phrases
7: in your file.
```

To replace all occurrences of `and` with `or` in the above file, type:

```
1,7Rand  or 
```

The resulting display is:

```
2: shows how the Replace commor works or how
4: certain words or phrases
6: other words or phrases
```

Lines 1, 3, 5, and 7 are not displayed because they are not changed.

To avoid unwanted changes, you can include the `?` parameter. For example, type:

```
2,7?Rand  or 
```

Now, whenever Replace finds the `and` string, you have the opportunity to accept or reject the change.

Note that in the first Replace command you type a space after `and` and `or`. In the second command, with `?R`, you do not type spaces.

Search Text (S)

[*line1*][,*line2*][?]S**[*string*]**

Searches all lines in the range *line1* to *line2* for each occurrence of the text string.

Notes

- You must terminate the string by pressing **ENTER**.
- If you omit the question mark (?), Search displays the first line (in the specified range) that matches the string. That line becomes the current line, and the Search command terminates.
- If the string does not occur between *line1* and *line2*, Search displays the message:

Not found

- If you include the question mark, Search displays the first line that contains the string. It then prompts you with the message:

O.K.?

If you press either **Y** or **ENTER**, the line becomes the current line, and the search terminates. If you press any other key, the search continues until it either finds another match or searches all lines.

- You can omit the *line1* parameter, as in:

,*line2*S***string* **ENTER****.

If you do, the search begins at the line that immediately follows the current line.

- You can omit the *line2* parameter, as in:

line1S***string* **ENTER**** or
line1*, **Sstring* **ENTER****

- If you omit the string, Search uses the last search string specified in a Replace or Search command. If there is no search string (no previous search or replace has been done), the command terminates immediately.
- Search looks for the string exactly as you specify it in the command. Therefore, enter the string in the same combination of upper and lowercase characters as it appears in the text.
- Begin the search string immediately after the S in the command line. End it by pressing **ENTER**. Any spaces you include are considered part of the search string.
- In multiple line commands, terminate the string with **CTRL Z**, instead of **ENTER**. You can then follow the string with another command beginning in the next character position.

Examples

Assume that the following file exists and is ready for editing. Line 5 is the current line.

```
1: This sample file
2: shows how the Search command functions
3: to locate and point out
4: a specified string
5: *in a range of lines
6: of your file.
7: The Search command
8: can include the optional parameter ? and
9: two line parameters for the range.
10: You then type the string and press ENTER
```

To search for the first occurrence of the string `and`, type:

```
1,10 SandENTER
```

The screen shows:

```
2: shows how the Search command functions
*__
```

because the string `and` is part of the word `command`. Line 2 then becomes the current line.

This is probably not the `and` you are looking for. To find the next occurrence of `and`, at the EDLIN prompt, type:

```
S 
```

The screen looks like this:

```
*1,10Sand
    2: shows how the Search command functions

*S
    3: to locate and point out
```

Line 3 becomes the current line.

To search through several occurrences of a string until you find the correct one, type:

```
1, ? Sand
```

The result is:

```
    2: shows how the Search command functions

O.K.? N
    3: to locate and point out

O.K.? N
    7: The Search command

O.K.? N
    8: can include the optional parameter ? and

O.K.? Y
```

The search continues until you press or the file runs out of lines. At that time, Search displays the message:

```
Not found
```

Transfer Lines (T)

[*line*]T[*drive*]*filename*

Inserts (merges) the contents of a specified file into the file being edited. The transferred file is inserted just ahead of the specified line or current line.

After the file is inserted, the merged file is renumbered automatically.

Notes

- If you omit *line*, then the file contents are inserted ahead of the current line.
- EDLIN reads the file to be transferred from the current directory of the specified drive or the default drive. If you issue a path when you start EDLIN, that path serves as the current directory. All subsequent Transfer Lines commands use that directory.

Example

```
10TB:myfile ENTER
```

inserts the contents of B:Myfile into the file being edited. B:Myfile is inserted just before Line 10.

Write Lines (W)

[*number*]W

Writes a specified number of edited lines from memory to disk. Writing begins with Line 1.

This command is meaningful only if the file you are editing is too large to fit into memory. When you start EDLIN, it loads the file into memory until memory is 75% full. To edit the rest of your file, you must first write the edited lines in memory to disk, using the Write Lines command. Then, you can load the unedited lines from disk into memory, using the Append command.

Note

- If you omit the number, EDLIN writes lines until 25% of memory is freed. If at least 25% already is freed, EDLIN takes no action. All lines remaining in memory (not written to disk) are renumbered, starting with Line 1.

Example

```
100W ENTER
```

writes Lines 1 through 100 to disk, and renumbers the file in memory, starting with Line 101 as Line 1.

LINKING OBJECT MODULES

MS-LINK™ is an MS-DOS utility program designed to produce ready-to-run machine language code. You should read all this chapter before you use the linker. It describes how to invoke the linker, and it summarizes options you can use with the linker. Chapters 12 and 13 contain additional information on the linker. The linker error messages are in Part 6.

If you do not expect to compile and link programs, you do not need to read Chapters 11, 12, and 13.

Basic Information

Programs are written in source code. By passing the source code through a compiler or an assembler, you produce an *object module*. This object module, however, cannot be understood by the computer directly.

To produce *machine language* code that the computer **can** understand, you must pass the object file, along with any required *library* files, through the link process.

MS-LINK processes object files generated by the Macro Assembler or by high-level-language compilers, such as C or Pascal. It copies the resulting program to an executable (.exe) output file. You can then run the program by typing the file's name at the MS-DOS system prompt.

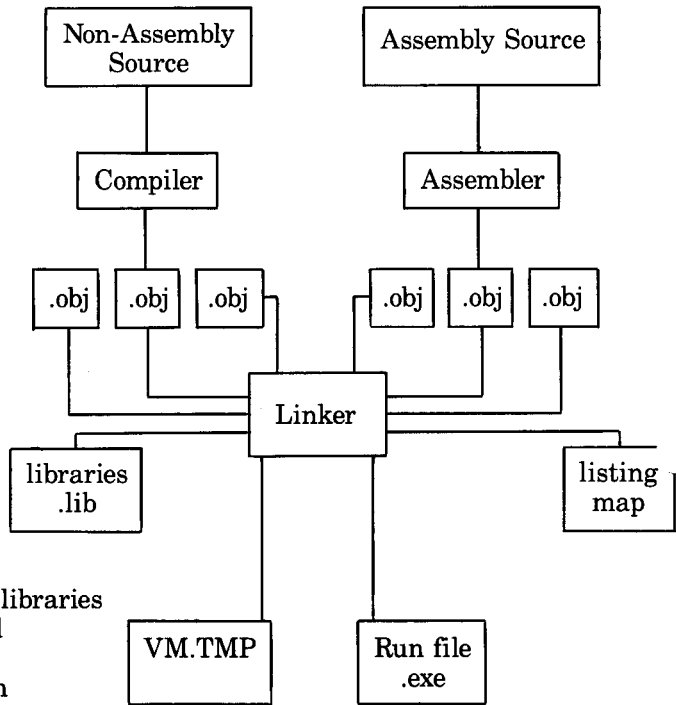
In addition to containing internal references, the object modules to be linked can contain *external references* (references to routines and variables defined in other modules). As it combines modules, the linker resolves all external references (makes sure they are defined). If an external reference is not defined in the object modules, the linker searches as many as 16 library files for the definition.

After resolving external references, the linker copies a relocatable execution image and relocation information to the executable file. Using the relocation information, MS-DOS can load the executable image at any convenient location and execute it.

The linker also produces a *map file* (also called a *listing file*) that shows external references resolved and displays error messages.

The linker can process programs that contain as much as one megabyte of code and data.

The following diagram illustrates the various parts of the linker's operation.



As many as 16 libraries
can be searched

Used only if run
file is larger than
memory

Starting and Using the Linker

You can use the linker in three different ways:

- By answering a series of prompts on the screen.
- By supplying an MS-DOS command line that contains the responses you would give the prompts.
- By supplying the name of a file that contains the responses you would give the prompts.

Once you start the linker, it either processes the files you supplied or prompts you for additional files. You can stop the linker at any time by pressing `CTRL C`.

Using the Prompt Method

Following is the general procedure for answering the linker's prompts. Additional details are given following the procedure.

1. To load the linker into memory and display its prompts, one at a time, type:

```
LINK ENTER
```

2. The linker prompts you for the object modules you wish to link, by displaying the following message:

```
Object Modules [OBJ]:
```

Type the name or names of the object modules. If you do not supply filename extensions, the linker uses .obj (as indicated in the prompt). If you use more than one name, separate the names with spaces or plus signs (+).

As with any MS-DOS command, be sure to specify an entire pathname for any file that does not exist in the current directory.

If you have more names than can fit on one line, type a plus sign as the last character on the line, and press `ENTER`. The linker prompts for additional object files.

After typing all object file names, press `ENTER`.

3. The linker prompts for the name of the executable file you want to create:

```
Run File [filename.EXE]:
```

Notice that the default filename (in brackets) is the same as the name entered at the `Object Modules` prompt.

If you want to supply a different filename for the executable file, type the name, and press `[ENTER]`. The linker supplies the extension `.exe` by default if you do not specify an extension.

If you want the linker to use the default filename, press `[ENTER]`. The filename will be the same as for the object file, except that it will have the extension `.exe`.

4. The next prompt is for the map (listing) file, which shows the external references resolved:

```
List File [NUL.MAP]:
```

Type the name of the map file you wish to create, and press `[ENTER]`. If you do not supply a filename extension, the linker uses `.map`. If you do not want a map file, simply press `[ENTER]`.

5. The last prompt is:

```
Libraries [.LIB]:
```

Type the names of any libraries containing routines or variables referred to but not defined in your program. If you give more than one name, separate the names with spaces or plus signs. If you do not supply extensions, the linker uses `.lib` by default.

If you have more names than can fit on one line, type the plus sign as the last character on the line, and press `[ENTER]`. The linker prompts for additional filenames.

After typing all names, press `[ENTER]`.

If you do not want to search any libraries, press `[ENTER]` without typing any names.

6. The linker creates the executable file. If the linker cannot find an object file, it displays a message and waits so you can change disks if necessary.

When answering the prompts described in the preceding procedure, you can use the linker options by typing them after the filename at any prompt. The linker options are described later in this chapter.

At any prompt, you can type the remaining filenames in the command line format described in the following section. For example, you can choose the default responses for all remaining prompts by entering a semicolon (;) after a prompt, or you can type commas (,) to indicate several files. When the linker encounters a semicolon, it immediately chooses the default responses, and processes the remaining files without displaying any more prompts.

Note: If you enter a semicolon at the `Object Modules` prompt, specify at least one object filename.

Example

```
LINK 
Object Modules [.OBJ]: moda+modb+ 
Object Modules [.OBJ]: modc+startup/PAUSE 
Run File [moda.EXE]: 
List File [NUL.MAP]: abc 
Libraries [.LIB]: B:\LIB\math 
```

The preceding example links the object modules `Moda.obj`, `Modb.obj`, `Modc.obj`, and `Startup.obj`. It searches the library file `Math.lib` in the `LIB` directory of Drive B for the routines and data used in the program. Then, it creates an executable file named `Moda.exe` and a map file named `Abc.map`. The `/PAUSE` option in the `Object Modules` prompt line causes the linker to pause so you can change disks. The linker then creates the executable file.

Using the Command Line Method

The second method of starting the linker is to supply a command line that lists the files you want the linker to process. The command has the following general format:

```
LINK object files [,executable file] [,map file] [,library files]]]
      [options] [;]
```

object files are the names of the object files you want to link. The files must be created using the assembler or a high-level compiler. You must supply at least one object-file name. If you omit the extension, the linker uses the `.obj` extension.

executable file is the name you wish to give the executable file that the linker creates. Specifying the name of the executable file is optional. If you omit the filename, the linker creates the name by using the filename of the first object file on the command line and appending to it the .exe extension.

map file is the name of the file to receive the map listing. This parameter is optional. You do not need to create a map file. If you specify a name for the map file, but you do not supply an extension, the linker appends the .map extension to the name you specify.

If you specify the /MAP or /LINENUMBERS option, the linker creates a map file even if you do not specify a map file in the command line.

library files is a list of the name(s) of the library files that contain routines you wish to link to create the program. This parameter is optional. Use it only if you have library files you need to link to the program. If you specify a filename of a library, but do not include an extension, the linker assumes the extension is .lib.

options are the available options listed in the “Linker Options” section of this chapter. You can use any of the options, and can put them anywhere on the command line.

The commas that separate the filenames in the command line are placeholders. You must include them even if you don’t supply the filenames. If you want a filename to be the default (for example, the base name of the first object file for the executable file), type the comma that would follow the filename (without supplying the filename).

You can use a semicolon anywhere after the object file to terminate the command line. If you type the semicolon after the object file, the linker supplies the default name for the executable file, and suppresses the map file and the library file.

If you do not supply all filenames in the command line, and you do not end the command line with a semicolon, the linker prompts for additional filenames, using the prompts described in the preceding section, “Using the Prompt Method.”

Consistent with the way MS-DOS accesses files, the linker assumes a file is in the current directory unless you specify otherwise. You cannot specify the drive and directory for the object file and expect the linker to supply the same drive and directory for other files. You must give the location of each file specifically.

Note: When linking modules produced by a high-level-language compiler that supports overlays, you must specify overlay modules by putting them in parentheses. The assembler has no overlay manager. However, you can use overlays with object modules compiled with FORTRAN (Version 3.2 or later), Pascal (Version 3.2 or later) and C (Version 3.0 or later). See your language compiler manual for details.

Examples

```
LINK file.obj,file.exe,file.map,routine.lib  
ENTER
```

is equivalent to:

```
LINK file,,file,routine ENTER
```

Either of the preceding command lines uses the file `File.obj` to create the executable file `File.exe`. Link searches the library file `File.lib` for the routines and variables needed. It also creates a file called `File.map`, which contains a list of the program's *segments* and *groups*. (See Chapter 12 for an explanation of segments and groups.)

```
LINK startup+file,B:file,\MAP\file; ENTER
```

uses the two object files, `Startup.obj` and `File.obj`, in the current directory, to create an executable file on Drive B. The linker creates a map file in the MAP directory of the current drive. It does not search any libraries.

```
LINK moda modb modc startup/PAUSE,,abc,  
B:\LIB\math ENTER
```

links the object modules `Moda.obj`, `Modb.obj`, `Modc.obj`, and `Startup.obj`. The linker searches the library file `Math.lib` in the LIB directory on Drive B for the necessary routines and data. It then creates an executable file named `Moda.exe`, and a map file named `Abc.map`. The PAUSE option causes the linker to pause so you can change disks before it creates the executable file. (See the "Linker Options" section, later in this chapter.)

Using the Response File Method

The final method for starting the linker is to create a program to do the job. You list, in a response file, the names of all the files to be processed, and then run the program by giving the name of the response file on the linker command line. The simplest form for the command is:

```
LINK @response-file-name
```

You can also specify a response file at any linker prompt, or at any position on the linker command line. The linker treats the input from the response file exactly as if you entered it at the prompts or in the command line. Placing a carriage return-line-feed combination in the response file has the same effect as pressing **ENTER** in response to a prompt or placing a comma in a command line.

You must precede the name of the response file with the symbol **@** to indicate that it refers to a response file, not to an object file.

Use a complete pathname if the response file is not in the current directory.

Name the file anything you like. The file has the following general form:

```
object files  
[executable file]  
[map file]  
[library files]
```

Place each group of filenames on a separate line. You can omit any elements already provided at prompts or with a partial command line. If you do not supply a filename for a group, leave an empty line.

If you have more names than can fit on one line, type a plus sign as the last character on the line. Then, continue typing names on the next line.

You can include options on any line. (See the “Linker Options” section, later in this chapter.)

You can also place a semicolon on any line in the response file. When the linker encounters the semicolon, it automatically supplies the default filenames for all files not yet named in the response file. It ignores the remainder of the response file.

When you create a program with a response file, the linker displays each response from the file on the screen in the form of prompts. If the file does not contain names for required files, the linker prompts for the missing names, and waits for you to enter responses.

Note: A response file should end with either a semicolon or a carriage return/linefeed combination. If you fail to provide the final CR/LF, the linker displays the last line of the response file, and waits for you to press

.

Examples

```
moda modb modc startup /PAUSE
abc
b:\LIB\math
```

The preceding response file tells the linker to link the four object files *Moda.obj*, *Modb.obj*, *Modc.obj*, and *Startup.obj*. The linker pauses to let you swap disks before producing the executable file *Moda.exe*. It also creates a map file, *Abc.map*, and searches the library *Math.lib* in the *\LIB* directory of Drive B.

The following procedure combines all three methods of supplying filenames. Assume you have a response file called *Library* that contains one line:

```
lib1+lib2+lib3+lib4
```

Now, start the linker with a partial command line:

```
LINK object1 object2 
```

The linker takes *Object1.obj* and *Object2.obj* as its object file, and prompts for the next file:

```
Run File [object1.EXE]: exec 
List File [NUL.MAP]: 
Libraries [LIB]: @library 
```

You enter *exec* so the linker will name the executable file *Exec.exe*. You then press to indicate that you don't want a map file. And, you enter *@library* so the linker will read the response file *Library*, which contains the names of the four library files.

Giving Search Paths With Libraries

You have two ways in which you can direct the linker to search for the libraries you want to link to a program:

- By specifying one or more *search paths* with the library names, either on the linker command line or in response to the Libraries prompt.
- By assigning the search paths to the environment variable LIB before you invoke the linker. You do this by using the MS-DOS SET command.

A search path is a drive name or the pathname of a directory. You can specify as many as 16 search paths. If you use the SET command to assign the paths, separate multiple paths with semicolons.

If you include a drive name or directory pathname when specifying a library on the linker command line, the linker searches only that specific location. If you don't give either of these (you use only a filename), the linker uses the following search order to find the library:

1. It searches the current drive and directory.
2. It uses the other search paths listed, in the order in which you gave them.
3. It uses any search path specified using the LIB environment variable.

If it still cannot find the file, the linker displays an error message.

Examples

```
LINK file,,file,A:\ALTLIB\math.lib+common+B:  
+D\LIB\ ENTER
```

causes the linker to search only the ALTLIB directory on Drive A for the library, Math.lib. However, to find Common.lib, the linker searches the current directory, the home directory of Drive B, and finally, the \LIB directory on Drive D.

```
SET LIB=C:\LIB;U:\SYSTEM\LIB   
LINK file,,file.map,math+common 
```

causes the linker to search the current directory, the \LIB directory on Drive C, and the SYSTEM\LIB directory on Drive U to find the libraries Math.lib and Common.lib.

The Map File

The map file lists the names, load addresses, and lengths of all segments in a program. It also lists the names and load addresses of any groups in the program, the program start address, and messages about any errors the linker might have encountered. If you use the /MAP option in the linker command line, the map file lists the names and load addresses of all public symbols.

Note: See Chapter 12 for information about groups, segments, and public symbols.

In the map file, segment information has the following general form:

Start	Stop	Length	Name	Class
00000H	0172CH	0172DH	TEXT	CODE
01730H	01E19H	006EAH	DATA	DATA

The Start and Stop columns show the 20-bit addresses (in hexadecimal) of the first and last byte in each segment. These addresses are relative to the beginning of the load module, which is assumed to be at address 0000 hex. The operating system chooses its own starting address when the program is loaded. The Length column gives the length of the segment (in bytes); the Name column, the name of the segment; and the Class column, the segment's class name.

Group information has the general form:

Origin	Group
0000:0	IGROUP
0173:0	DGROUP

In this example, IGROUP is the name of the code (instruction) group, and DGROUP is the name of the data group.

At the end of the listing file, the linker gives you the address of the program entry point.

If you specify the `/MAP` option in the linker command link, the linker adds a public-symbol list to the map file. It presents the symbols twice—first in alphabetical order, then in order of their load addresses. The public-symbol list has the following general form:

Address	Publics by Name
0000:1567	BRK
0000:1696	CHMOD
0000:01DB	CHKSTK
0000:131C	CLEARERR
0173:0035	FAC

Address	Publics by Value
0000:01DB	CHKSTK
0000:131C	CLEARERR
0000:1567	BRK
0000:1696	CHMOD
0000:0035	FAC

The addresses of the public symbols are in *segment:offset* format. They show the location of the symbols relative to the beginning of the load module, which is assumed to be at address 0000:0000.

If you use the `/HIGH` and `/DSALLOCATE` options, and the program's code and data combined do not exceed 64K bytes, the map file might show symbols that have unusually large segment addresses. These addresses indicate a symbol that is located below the actual start of program code and data. For example, the symbol entry:

```
FFF0:0A20    TEMPLATE
```

shows that `TEMPLATE` is located below the start of the program. Note that the 20-bit address of `TEMPLATE` is 00920 hex.

The Temporary Disk File: VM.TMP

Normally, the linker uses available memory for its session. However, if it runs out of available memory, it creates a temporary disk file named VM.TMP in the current directory. When it creates this file, the linker displays:

```
VM.TMP has been created.  
Do not change diskette in drive x
```

After this message appears, do not remove the disk from the specified drive until the link session ends. The /PAUSE option cannot be used if the linker has created a temporary file.

After the linker creates the executable file, it automatically deletes the temporary file.

Note: Do not use the filename VM.TMP for your own files. When the linker creates the temporary file, it destroys any previous file that has the same name.

Using Linker Options

The linker options let you specify and control tasks you want the linker to perform. All options begin with a the linker-option character, which is the forward slash (/). You can use an option anywhere on a linker command line.

The following list summarizes the linker options:

OPTION	PURPOSE
/PAUSE	Cause the linker to pause during linking
/EXEPACK	Packs the executable file
/MAP	Creates a public symbol map
/LINENUMBERS	Copies line numbers to the map file
/NOIGNORECASE	Preserves case sensitivity in names
/NODEFAULTLIBRARYSEARCH	Overrides default libraries
/STACK	Sets the stack size
/CPARMAXALLOC	Sets the maximum allocation space
/HIGH	Sets the high load address
/DSALLOCATE	Allocates the data group
/NOGROUPASSOCIATION	Sets the group association override
/OVERLAYINTERRUPT	Sets the overlay interrupt
/SEGMENTS	Sets the maximum number of segments
/DOSSEG	Specifies MS-DOS segment ordering

You can abbreviate option names, as long as your abbreviations contain enough letters to distinguish each option from the others. The minimum abbreviation for each option is listed below each option name in the sections that follow.

Pausing to Change Disks

/PAUSE
/P

The /PAUSE option causes the linker to pause so you can swap disks before it writes the executable file to disk. If you specify /PAUSE, the linker displays:

```
About to generate .EXE file
Change diskette in drive x and press ENTER
```

The message appears after the linker reads data from the object and library files and after it writes data to the map file, if you specified one. The linker resumes processing after you press **ENTER**. After writing the executable file, the linker displays:

```
Please replace original diskette
in drive x and press ENTER
```

Note: If the linker creates a VM.TMP file, do not remove the disk that file is on. If the temporary disk message appears when you have specified the /PAUSE option, press **CTRL C** to terminate the link session. Then, rearrange your files so the linker can write the temporary file and the executable file to the same disk. Start the linker again.

Example:

```
LINK file/PAUSE,file,,\LIB\math ENTER
```

causes the linker to pause immediately before creating the executable file, File.exe. After creating the file, the linker pauses again to let you replace the original diskette.

Packing Executable Files

```
/EXEPACK
/E
```

The /EXEPACK option directs the linker to remove sequences of repeated bytes (typically nulls) and to optimize the load-time relocation table before creating the executable file. Executable files linked with this option might be smaller, and, therefore, might load faster than files linked without the option.

The /EXEPACK option does not always reduce the file size significantly. Sometimes, it even increases the size. Generally, programs that have a large number of load-time relocations (approximately 500 or more) and long streams of characters are shorter if packed. If you're not sure if your program meets these conditions, try linking it both ways, and compare the results.

Example:

```
LINK program /E; ENTER
```

creates a packed version of the executable file Program.exe.

Producing a Public-Symbol Map

/MAP
/M

The /MAP option causes the linker to produce a listing of all the public symbols declared in your program. The linker copies the list to the map file it creates. For a complete description of the map file format, see “The Map File,” earlier in this chapter.

If you do not specify a map file in the linker command, you can use the /MAP option to force the linker to create a map file. The linker gives the forced map file the same filename as the first object file specified in the command. It appends the default extension /MAP.

Example:

```
LINK file,,/MAP; ENTER
```

creates a map of all public symbols in the object module File.obj.

Copying Line Numbers to the Map File

/LINENUMBERS
/LI

The /LINENUMBERS option directs the linker to copy the starting address of each program source line to a map file. The starting address is the address of the first instruction that corresponds to the source line.

The linker copies the line number data only if you give a map file name in the linker command line, and only if the given object file has line number information. Otherwise, it ignores the option.

Line numbering is available in some high-level-language compilers, including FORTRAN and Pascal (Versions 3.0 and later) and C (Version 2.0 and later). The Macro Assembler does not have line numbering information in its object files.

Note: If you do not specify a map file in a linker command, you can still use the /LINENUMBERS option to force the linker to create a map file. To do this, place the option at or before the List File prompt. The linker assigns the forced map file the same name as the first object file specified in the command line, and appends the default extension, .map.

Example:

```
LINK file/LINENUMBERS,,,em+slibfp ENTER
```

causes the linker to copy information about the line numbers in the object file File.obj to the map file File.map.

Preserving Lowercase

```
/NOIGNORECASE  
/NOI
```

The `/NOIGNORECASE` option causes the linker to treat upper- and lowercase letters in symbol names as distinct letters. Normally, the linker considers upper- and lowercase letters to be identical, for example, treating the names `TWO`, `two`, and `Two` as the same symbol. When you use `/NOIGNORECASE`, the linker treats the names as different symbols.

Typically, you use the `/NOIGNORECASE` option with object files created by high-level-language compilers. Some compilers treat upper- and lowercase letters as distinct letters and assume the linker does the same.

If you are linking modules created with the Macro Assembler to modules created with a case-sensitive language, such as C, be sure public symbols have the same sensitivity in both modules. For example, you can make all variables in C distinctive by spelling, regardless of case, and then link without the `/NOIGNORECASE` option. Another alternative is to use the `/ML` or `/MX` option to make public variables in the Macro Assembler case-sensitive, and then link with the `/NOIGNORECASE` option.

Example:

```
LINK file1+file2/NOI,,,em+mlibfp ENTER
```

causes the linker to treat upper- and lowercase letters in symbol names as distinct letters. The linker links the object files, File1.obj and File2.obj, with subroutines from the standard C language libraries, Em.lib and Mlibfp.lib, located in the \LIB directory. The C language expects upper and lower-case letters to be treated as distinct.

Ignoring Default Libraries

```
/NODEFAULTLIBRARYSEARCH  
/NOD
```

The `/NODEFAULTLIBRARYSEARCH` option directs the linker to ignore any library names it might find in an object file. A high-level-language compiler might add a library name to ensure that the linker links a default set of libraries with the program. Using the `/NODEFAULTLIBRARYSEARCH` option overrides such default libraries, and lets you name the libraries you want by including them on the linker command line.

Example:

```
LINK startup+file/NOD,,,em+slibfp+slibc 
```

links the object files, `Startup.obj` and `File.obj`, with the routines from the libraries `Em.lib`, `Slibfp.lib`, and `Slibc.lib`. The linker ignores any libraries that might be named in the object files.

Setting the Stack Size

```
/STACK:size  
/ST:size
```

The `/STACK` option sets the program stack to the number of bytes specified by *size*. Normally, the linker calculates a program's stack size automatically, basing it on the size of any stack segments given in the object files. If you use the `/STACK` option, the linker uses the specified *size* instead of a value it might have calculated.

size can be any positive integer value in the range 1 to 65535. The value can be a decimal, octal, or hexadecimal number. Begin octal numbers with a zero. Begin hexadecimal numbers with a leading zero followed by a lowercase x (for example, `0x1B`).

Examples:

```
LINK file/STACK:512,,,; 
```

sets the stack size to 512 bytes.

```
LINK moda+modb,run/ST:0xFF,ab,\LIB\start; 
```

sets the stack size to 255 (FF hex) bytes.

```
LINK startup+file/ST:030,;; 
```

sets the stack size to 24 (30 octal) bytes.

Setting the Maximum Allocation Space

```
/CPARMAXALLOC:number  
/C:number
```

The /CPARMAXALLOC option sets the maximum number of 16-byte paragraphs needed by the program when it is loaded into memory. The operating system uses this number when allocating space for a program prior to loading it.

Normally, the linker sets the maximum number of paragraphs to 65535. Because this represents all addressable memory, the operating system always denies the request, and allocates the largest contiguous block of memory it can find. If you use the /CPARMAXALLOC option, the operating system allocates no more space than you specify. Any additional space in memory is free for other programs.

number can be any positive integer value in the range 1 to 65535. The value can be a decimal, octal, or hexadecimal number. Begin octal numbers with a zero. Begin hexadecimal numbers with a leading zero followed by a lowercase x (for example, 0x2B).

If *number* is less than the minimum number of paragraphs the program needs, the linker ignores your request, and sets the maximum value equal to the minimum needed. The minimum number of paragraphs needed is never less than the number of paragraphs of code and data in the program.

Examples:

```
LINK file/C:15,;; 
```

sets the maximum allocation to 15 paragraphs.

```
LINK moda+modb,run/CPARMAXALLOC:0xFF,ab; 
```

sets the maximum allocation to 255 (FF hex) paragraphs.

```
LINK startup+file,/C:030,;; 
```

sets the maximum allocation to 24 (30 octal) paragraphs.

Setting a High Start Address

/HIGH
/H

The /HIGH option sets the program's starting address to the highest possible address in free memory. If you do not use this option, the linker sets the starting address as low as possible in memory.

Example:

```
LINK startup+file/HIGH,;, 
```

sets the starting address of the program in File.exe to the highest possible address in free memory.

Allocating a Data Group

/DSALLOCATE
/D

The /DSALLOCATE option directs the linker to reverse its normal processing when assigning addresses to items belonging to the group named DGROUP. Normally, the linker assigns the offset 0000 hex to the lowest byte in a group. If you use /DSALLOCATE, the linker assigns the offset FFFF hex to the highest byte in the group. The result is data that appears to be loaded as high as possible in the memory segment containing DGROUP.

Typically, you use the /DSALLOCATE option with the /HIGH option to take advantage of unused memory before the start of the program. The linker assumes that all free bytes in DGROUP occupy the memory preceding the program. To use the group, you must set a segment register to the start address of DGROUP.

Example:

```
startup+file/HIGH/DSALLOCATE,;,em+m1ibfp 
```

directs the linker to place the program as high as possible in memory, and then to adjust the offsets of all data items in DGROUP so they are loaded as high as possible within the group.

Removing Groups From a Program

`/NOGROUPOSSOCIATION`
`/NOG`

The `/NOGROUPOSSOCIATION` option directs the linker to ignore group associations when assigning addresses to data and code items.

This option exists strictly for compatibility with older versions of FORTRAN and Pascal (Microsoft Versions 3.13 or earlier, or any IBM version prior to 2.0). Use the `/NOGROUPOSSOCIATION` option **only** if you are linking with object files produced by those compilers or with runtime libraries that accompany the old compilers.

Example:

```
startup+file/NOG,,,em+mlibfp ENTER
```

directs the linker to ignore group associations when assigning addresses to data and code items.

Setting the Overlay Interrupt

`/OVERLAYINTERRUPT:number`
`/O:number`

The `/OVERLAYINTERRUPT` option sets the interrupt number of the overlay loading routine to the specified *number*. This option overrides the normal overlay interrupt number (03F hex).

number can be any integer value in the range 0 to 255. The value can be a decimal, octal, or hexadecimal number. Begin octal numbers with a zero. Begin hexadecimal numbers with a leading zero followed by a lowercase x (for example, 0x3B).

You can use the `/OVERLAYINTERRUPT` option only if you are linking with a runtime module from a language compiler that supports overlays. Check your compiler documentation to see if this option is appropriate for your compiler. The Macro Assembler does not have an overlay manager. Therefore, you cannot use the `/OVERLAYINTERRUPT` with modules assembled with it.

Note: Do not use interrupt numbers that conflict with the standard MS-DOS interrupts.

Examples:

```
LINK file/0:255,,,87+slibfp 
```

sets the overlay interrupt number to 255.

```
LINK moda+modb,run/OVERLAY:0xff,ab.map,em+mllibfp 
```

sets the overlay interrupt number to 255 (FF hex).

```
LINK startup+file,/0:0377,,em+mllibfp 
```

sets the overlay interrupt number to 255 (377 octal).

Setting the Maximum Number of Segments

/SEGMENTS:number

/SE:number

The */SEGMENTS* option instructs the linker to process no more than the specified *number* of segments per program. If the linker encounters more than the given limit, it displays an error message and stops linking.

If you do not use */SEGMENTS*, the linker allocates enough memory space to process as many as 128 segments. If your program has more than 128 segments, you need to set the segment limit higher.

If the linker displays the following error message:

```
Segment limit set too high
```

you need to set the segment limit lower.

number can be any value in the range 1 to 1024. The value can be a decimal, octal, or hexadecimal number. Begin octal numbers with a zero. Begin hexadecimal numbers with a leading zero followed by a lowercase x (for example, 0x4B).

Examples:

```
LINK file/SE:192,,, 
```

sets the segment limit to 192.

```
LINK moda+modb,run/SEGMENTS:0xff,ab,em+mllibfp; 
```

sets the segment limit to 255 (FF hex).

Using DOS Segment Order

/DOSSEG
/DO

The /DOSSEG option causes the linker to arrange all segments in the executable file according to the MS-DOS segment-ordering convention. This convention adheres to the following rules:

- All segments that have the class name CODE are placed at the beginning of the executable file.
- Any other segments that do not belong to the group named DGROUP are placed immediately after the CODE segments.
- All segments belonging to DGROUP are placed at the end of the file.

See “Order of Segments,” in Chapter 12, for an explanation of the normal segment order (used if you do not specify /DOSSEG).

Example:

```
LINK start+test/DOSSEG,,,math+common 
```

causes the linker to create an executable file, File.exe, that has its segments arranged according to the MS-DOS segment-ordering convention. The segments in the object files, Start.obj and Test.obj, and any segments copied from the libraries, Math.lib and Common.lib, are arranged as specified above.

HOW THE LINKER WORKS

This chapter is to help you understand how the linker works. Generally, if you are linking object modules compiled from BASIC, Pascal, or another high-level language, you do not need this information. If you are writing and compiling programs in assembly language, however, you do need it.

Segments, Classes, and Groups

In MS-DOS, memory can be divided into segments, classes, and groups.

A *segment* is a contiguous area of memory a maximum of 64K bytes long. The contents of a segment are addressed by a *segment address* and an *offset* within that segment. Segments can overlap.

A *group* is a collection of segments that fit within a 64K byte area of memory. The segments within the group do not need to be contiguous.

The lowest address of the lowest segment in a group is called the *group address*. This is the address that the linker uses to reference the segments in the group. The starting address of the first segment in a group is called the *canonical frame number*. This is the address from which offsets are calculated.

A program can consist of one or more groups. When writing in assembly language, you can assign the names of the groups in your program. In high-level languages, the compiler automatically assigns the names.

A *class* is a collection of segments. The naming of segments to a class controls the order and relative placement of segments in memory. In an assembly-language program, you assign names to the classes. In high-level language programs, the compiler automatically assigns the names.

Notice, from the following example, that segments must have unique segment names, but they can have duplicate class names (because more than one segment can belong to one class):

	Segment Name	Class Name
Segment 1	PROG.1	CODE
Segment 2	PROG.2	CODE
Segment 12	PROG.3	DATA

How the Linker Uses Segments, Classes, and Groups

The linker creates an executable file by concatenating a program's *code* and *data* segments according to the instructions in the original source files. It forms the concatenated segments into an executable image, which is copied directly into memory when you invoke the program for execution. **Therefore, the order and manner in which the linker copies segments to the executable file defines the order and manner in which it loads the segments into memory.**

You can tell the linker how to load a program's segments by using two directives:

- The `SEGMENT` directive, to supply segment attributes.
- The `GROUP` directive, to form segment groups.

These directives define group associations, classes, and *align* and *combine* types—which, in turn, define the order and relative starting addresses of all segments in a program. The information you supply through the directives works in addition to any information you supply through command line options.

The rest of this chapter explains the process that the linker uses to concatenate segments and to resolve references to items in memory.

Alignment of Segments

The *align* type is the segment attribute that the linker uses to set the starting address for the segment. In an assembly-language program, **you** specify the align type.

- *Byte* alignment tells the linker it can start a segment on any byte boundary (that one segment can immediately follow another.)
- *Word* alignment tells the linker to start segments only on word boundaries (addresses that are multiples of 2).
- *Para* alignment tells the linker to start segments only on paragraph boundaries (addresses that are multiples of 16).
- *Page* alignment tells the linker to start segments at page boundaries (addresses that are multiples of 256).

The default align type is para.

When the linker encounters a segment, it checks the align type before copying the segment to the executable file. If the align type is word, para, or page, the linker checks the executable image to see if the last byte copied ends at an appropriate boundary. If not, the linker pads the image with extra null bytes.

Canonical Frame Number

The linker computes a starting address for each segment in a program. It bases this address on the segment's align type and the size of the segments already copied to the executable file. The address consists of an offset and a canonical frame number, which specifies the address of the first paragraph in memory that contains one or more bytes of the segment.

A frame number is always a multiple of 16 (a paragraph address). The offset is the number of bytes from the start of the paragraph to the first byte in the segment.

For the byte and word align types, the offset may be nonzero. The offset is always zero for the para and page align types.

The frame number of a segment can be obtained from a linker file. The frame number is the first five hexadecimal digits of the start address specified for the segment.

Order of Segments

The linker copies segments to the executable file in the order in which it encounters them in the object files. It maintains this order throughout the program unless it encounters two or more segments with the same class name. Segments with the same class name belong to the same class type. Therefore, the linker copies them to the executable file as contiguous blocks.

Combined Segments

The linker uses *combine types* to determine whether it should combine multiple segments having the same segment name into a single, large segment. The combine types are public, common, memory, at, and private.

If a segment's combine type is *public*, the linker automatically combines the segment with any other segments that have the same name and that belong to the same class. When the linker combines segments, it ensures that the segments are contiguous and that all addresses in the segments can be accessed using an offset from the same frame address. The result is the same as if the segment were defined as a whole in the source file.

The linker preserves each segment's align type. This means that even though the segments belong to a single, large segment, the code and data in the segments retain their original align type. If the combined segments exceed 64K bytes, the linker displays an error message.

If a segment's combine type is *stack*, the linker carries out the same combine operation as for public segments. The only difference is that the stack segments cause the linker to copy an initial stack-pointer value to the executable file. This stack-pointer value is the offset to the end of the first stack segment (or combined stack segment) that the linker encounters. If you use the stack type for stack segments, you do not need to give instructions that load the segment into the SS register.

If a segment's combine type is *common*, the linker automatically combines it with any other segments that have the same name and that belong to the same class. When the linker combines common segments, however, it places the start of each segment at the same address, creating a series of overlapping segments. The result is a single segment that is no larger than the largest of the combined segments.

The linker treats segments with the *memory* combine type exactly like segments with the *public* combine type. The Macro Assembler provides the *memory* type for compatibility with linkers that support a separate combine type for memory segments.

A segment has the *private* combine type only if no explicit combine type is defined for it in the source file. The linker does not combine private segments.

Groups

Groups permit non-contiguous segments that do not belong to the same class to be addressable relative to the same frame address. When the linker encounters a group, it adjusts all memory references to items in the group so that they are relative to the same frame address.

Segments in a group do not have to be contiguous, do not have to belong to the same class, and do not have to have the same combine type. The only requirement is that all segments in the group fit within 64K.

Groups do not affect the order in which the segments are loaded. Unless you use class names and enter object files in the proper order, there is no guarantee that the segments will be contiguous. In fact, the linker might place segments that do not belong to the group in the same 64K bytes of memory. Although the linker does not explicitly check to see that all segments in a group fit within 64K of memory, it is likely to encounter a fixup-overflow error if this requirement is not met.

Fixups

Once the linker knows the starting address of each segment in a program, and all segment combinations and groups are established, the linker can fix any unresolved references to labels and variables. To fix unresolved references, the linker computes an appropriate offset and segment address, and replaces the temporary values, generated by the assembler, with the new values.

The linker carries out fixups for four types of references:

- Short
- Near self-relative
- Near segment-relative
- Long

The size of the value to be computed depends on the type of reference. If the linker discovers an error in the expected size of a reference, it displays a `fixup-overflow` message. This can happen, for example, if a program attempts to use a 16-bit offset to reach an instruction in a segment that has a different frame address. It can also occur if all segments in a group do not fit within a single 64K block of memory.

A *short* reference occurs in `JMP` instructions that attempt to pass control to labeled instructions that are in the same segment or group. The target instruction must be no more than 128 bytes from the point of reference. The linker computes a signed, 8-bit number for this reference, and displays an error message if the target instruction belongs to a different segment or group (has a different frame address), or if the target is more than 128 bytes distant (in either direction).

A *near self-relative* reference occurs in instructions that access data relative to the same segment or group. The linker computes a 16-bit offset for this reference, and displays an error message if the data items are not in the same segment or group.

A *near segment-relative* reference occurs in instructions that attempt to access data that is in a specified segment or group, and that is relative to a specified segment register. The linker computes a 16-bit offset for this reference, and displays an error message if the offset of the target within the specified frame is greater than 64K or less than 0, or if the beginning of the canonical frame of the target is not addressable.

A *long* reference occurs in `CALL` instructions that attempt to access an instruction in another segment or group. The linker computes a 16-bit frame address and a 16-bit offset for this reference. It displays an error message if the computed offset is greater than 64K or less than 0, or if the beginning of the canonical frame of the target is not addressable.

USING THE LIBRARY MANAGER

MS-LIB™, the library manager, is provided for advanced programmers. With the library manager, you can create library files to use with the linker. You can also modify library files by:

- Deleting modules from a library.
- Adding object files (as modules) to a library.
- Replacing modules. To do this, first use the delete function and then the add function.

In addition, you can *extract* a module from a library file and place it in a separate object file. Extraction does not delete the module from the library; it copies it.

Order of Operations

During each library session, the library manager first deletes or extracts modules. Then, it appends new modules to the end of the file. During these operations, the library manager reads each module into memory, and checks it for consistency. It then writes back to the file all modules you want to retain. While doing so, it closes up the disk space to keep the library file as small as possible.

After appending all new modules, the library manager creates the index that the linker uses to find modules and symbols in the library file. If you wish, you can instruct the library manager to store the index in a map file. The file contains two lists. The first is an alphabetical list of all PUBLIC symbols, each followed by the name of the module that contains it. The second is a cross-reference list—an alphabetical list of the modules, each followed by a list of the PUBLIC symbols in the module.

Starting and Using the Library Manager

You can start the library manager in the same three ways that you can start the linker:

- By answering a series of prompts on the screen.
- By supplying an MS-DOS command line that contains the responses you would give the prompts.
- By supplying the name of a file that contains the responses you would give the prompts.

Several command characters help simplify the task of using the library manager. They are discussed later in “Command Characters.” You might want to refer to that section when reading about the different methods for starting the library manager.

Using the Prompt Method

Following is the general procedure for answering the library manager's prompts. Additional details are given following the procedure.

1. To load the library manager into memory and display its prompts, one at a time, type:

```
LIB 
```

2. The library manager prompts you for the library you want to create or modify:

```
Library File:
```

Enter the name of the library file. Omit the extension if you want the library manager to assume the default extension (.lib). For example to specify the library Sample.lib, type the following at the prompt:

```
sample 
```

If you specify a library that does not exist, the library manager displays:

```
Library file does not exist. Create?
```

Type YES to create the file or NO to stop the session.

3. The linker prompts you for the operation(s) you want to perform on the file:

Operation:

List, in any order, any modules you want to delete or extract and any object files you want to append. Precede each name with the command character that specifies the type of operation you want to perform on that module or file. The command characters that apply are:

- Minus sign (-), to delete
- Asterisk (*), to extract
- Plus sign (+), to append

For example, to delete the module Less and append the Object file More, type:

```
+more-less 
```

Default drive specifications for the Operation prompt vary with the type of operation. See “Command Characters” for a detailed explanation of the command characters and defaults.

4. The linker prompts for the name of a map (listing) file to create:

List file:

If you want a map file, enter the filename. For example to create the map file Crosslst, type:

```
crosslst 
```

If you do not want a map file, press only . The library manager uses NUL and thus does not create a map file.

Using the Command Line Method

The second method of starting the library manager is to supply a command line that lists the library and the operations you want to perform. The command has the following general format:

LIB library operations,listing

The command line options, defined under “Using the Prompt Method,” accomplish the same purposes outlined in that section. For example, the following command deletes the module Heap from the library Pascal.lib, and then appends the object file Heap.obj to that library:

```
LIB pascal-heap+heap 
```

Notice that there is no space between the library and the first command character.

If you type only a library name followed by a semicolon (;), the library manager reads the library file and checks it for consistency. For example, to perform a consistency check on the Pascal library file, type:

```
LIB pascal; 
```

If you type only a library name followed by a comma (,) and a map file name, the library manager checks the library file for consistency, and produces the map file. It performs no other operations. For example, to perform a consistency check on the library file Pascal and to create the map file Pascross.pub, type:

```
LIB pascal,pascross.pub 
```

Using the Response File Method

The final method for starting the library manager is to create a program to do the job. You list, in a response file, the same information you would supply using the prompt method. The simplest form for the command is:

```
LIB @response-file-name
```

The library manager treats the input from the response file exactly as if you entered it at the prompts or in the command line. Placing a carriage return-linefeed combination in the response file has the same effect as pressing in response to a prompt or placing a comma in a command line.

You must precede the name of the response file with the symbol @ to indicate that it refers to a response file, not to a library file.

Name the file anything you like. Use a complete pathname if the response file is not in the current directory.

A response file has one line of text for each response. Be sure your responses are in order so they apply to the appropriate prompts.

Use the command characters in the response file the same as you use them at the keyboard.

When the library session begins, the library manager displays each prompt with its response.

Examples:

```
pascal   
,crosslst 
```

The preceding response file causes the library manager to read the library file Pascal.lib, perform a consistency check, and produce the map file Crosslst.

```
pascal   
+more-less 
```

The preceding response file causes the library manager to delete the module Less from the library file Pascal.lib and to add the object file More.obj. No listing file is created.

Command Characters

Several command characters help simplify the task of using the library manager. They are:

- The plus sign (+). Use a plus sign preceding an object filename to instruct the library manager to append that object file as a module in the specified library.

When the library manager does the append, it removes the drive specification and extension from the object file specification. For example, the object file B:Cursor.obj becomes the module Cursor.

- The minus sign (-). Use a minus sign preceding a module name to instruct the library manager to delete that module.
- The asterisk (*). Use an asterisk preceding a module name to instruct the library manager to extract (copy) that module from the library into an object file.

The library manager assigns the object file specification, using this format:

current drive:module name.obj

For example, if the current drive is Drive A, and the module name is Cursor, the object file specification is A:Cursor.obj. If you do not want to use the current drive but not the .obj extension, rename the file.

- The ampersand (&). Use this symbol to extend the current line when you specify the operations to perform. When you place an ampersand at the end of a line, the library manager displays the Operation prompt again so that you can type more responses:

```
Operation: +cursor-heap+heap*foibles&  
Operation: *int+assume+ride; 
```

Use the ampersand as many times as necessary. Only disk space limits the number of modules you can append or extract. You can delete as many modules as exist.

- The semicolon (;). You can use the semicolon at any time after the first prompt (Library File) to select default responses to the remaining prompt(s). To do so type ; .

Caution: Once you enter the semicolon, you can no longer respond to any of the remaining prompts. Therefore, do not use the semicolon to skip only one prompt. To skip only one prompt, use the key.

- . Pressing stops the library session at any time. If you enter an incorrect response, such as an incorrect filename or module name, press to exit the library manager. Then, restart the session.

If you make an error before you press , use to delete characters in that line.

STARTING DEBUG

The MS-DOS DEBUG is a utility program to aid you in testing, changing, and observing the operation of executable object files. You can use DEBUG to alter the contents of a file or register and then immediately execute the program to see if the changes are valid. You do not have to reassemble the program.

To enter the DEBUG program, type:

DEBUG [*pathname* [*parameters*]]

- *pathname* specifies the program file that you want to load into memory. DEBUG loads the file, starting at 100H in the lowest available segment. The BX:CX registers are loaded with the number of bytes placed into memory.

Omitting the *pathname* lets you work with the current register contents. To load a file into memory, you must use the NAME and LOAD commands.

Note: When DEBUG is initialized, it sets a program segment prefix (PSP) at Offset 0 in the program work area. If you don't specify a *pathname*, you may overwrite the default PSP. If you are debugging a .com or .exe file, however, tampering with the PSP below relative address 5CH causes DEBUG to terminate.

- *parameters* is a list of *pathname* parameters and switches that are to be passed to the program when it is loaded. Include parameters only if you include a *pathname*.

A common use of parameters is to enable you to debug your program under runtime conditions. For example if you have a fast file-copying program, you can type:

DEBUG fastcopy myfile hisfile

where Fastcopy is the program you are debugging, Myfile is the file you want to copy, and Hisfile is the file you want to copy to.

DEBUG displays a hyphen (-) to indicate that it is ready to accept a command.

At initialization, the DEBUG segment registers (CS, DS, ED, and SS) are set to the bottom of free memory, and the instruction pointer (IP) is set to 0100H. All flags are cleared, and the remaining registers are set to zero.

Do not restart a program after DEBUG displays the message `Program terminated normally`. To reload the program, use the NAME and LOAD commands. Otherwise, the program does not run properly.

INTRODUCTION TO DEBUG REFERENCE

A **DEBUG** command consists of one letter followed by one or more parameters. You can use any combination of uppercase and lowercase letters in commands and parameters. While using **DEBUG**, you can also use the control keys and the MS-DOS editing functions (described in Part 3).

Notes

- You can terminate any **DEBUG** command by pressing **CTRL** **C**.
- To stop the screen from scrolling, press **CTRL** **S**. To continue scrolling, press **CTRL** **S** again.
- If a syntax error occurs in a **DEBUG** command, **DEBUG** displays the command line with the error indicated by ↑ and the word **error**. For example:

```
D CS:900 CS:110
    ↑ Error
```

Summary of DEBUG Commands

Command	Syntax
ASSEMBLE	A [<i>address</i>]
COMPARE	C <i>range address</i>
DUMP	D [<i>address</i>] D [<i>range</i>]
ENTER	E <i>address [list]</i>
FILL	F <i>range list</i>
GO	G [= <i>address1</i> [<i>address2...</i>]]
HEX	H <i>value1 value2</i>
INPUT	I <i>portaddress</i>
LOAD	L [<i>address [drive sector sectorcount]</i>]
MOVE	M <i>range address</i>
NAME	N <i>filespec1 [filespec2]</i>
OUTPUT	O <i>portaddress byte</i>
PROCEED	P [= <i>address</i>] [<i>value</i>]
QUIT	Q
REGISTER	R [<i>registername</i>]
SEARCH	S <i>range list</i>
TRACE	T [= <i>address</i>]/[<i>value</i>]
UNASSEMBLE	U [<i>address</i>] U [<i>range</i>]
WRITE	W [<i>address [drive sector sectorcount]</i>]

Command Parameters

All DEBUG commands except QUIT accept parameters. Parameters can be separated with spaces or commas. This is required only between two consecutive hexadecimal values. Thus, the following commands are the same.

```
DCS:100 110
D CS:100 110
D,CS:100,110
```

address is a 1- or 2-part designation in one of the following formats.

- An alphabetic segment register designation and an offset value. Example: CS:0100
- A 4-digit segment address and an offset value. Example: 04BA:0100
- An offset only, in which case the default segment is used. DS is the default segment for all commands except GO, LOAD, TRACE, UNASSEMBLE, and WRITE, for which the default segment is CS.

All numeric values are hexadecimal. You must use a colon to separate a segment designation and an offset.

byte is a 1- or 2-character hexadecimal value to be placed in or read from an address or register.

drive is a 1-digit value indicating which drive is to be used for accessing or writing data.

```
0 = Drive A
1 = Drive B
2 = Drive C
3 = Drive D
```

filespec is a file specification consisting of a drive specification, filename, and filename extension. (The three fields are optional, but at least the drive specification or filename you should include.)

list is a series of strings or byte values. *list* must be the last command line parameter. Example:

CS:100 FF 42"XXX" 1A 3

portaddress is a hexadecimal value that specifies a port number. It can have a maximum of four characters.

range is an area of memory, specified by either of these formats:

- *address1 address2*

Example: CS:100 110

address2 must be an offset value.

- *addressLvalue*

value is the number of bytes on which a command operates. If you omit *Lvalue*, DEBUG assumes a value of 80 bytes. Do not use this format if another hex value follows the range because the hex value would be interpreted as the second address of the range.

Example: CS:100 L 10

CS:100

The minimum value for *range* is 10000 hex. To specify a value of 10000 hex within 4 digits, enter 0000 (or 0).

registername (See the explanation of the REGISTER command for a list of valid register names.)

sector sectorcount is a 1- to 3-character hexadecimal value indicating the relative sector number on the disk and the number of disk sectors to be written or loaded.

The first sector of Track 0 on Head 0 is Sector 0. The remaining sectors of Head 0, Track 0 are numbered consecutively. Numbering continues with the first sector of the Head 1 track that is of the same radius as Track 0. When all sectors on all heads of the track (that is of the same radius as Track 0) are numbered, numbering continues with the first sector of Head 0 of the next track.

string is any number of characters enclosed in quotation marks, either single (') or double ("). The ASCII values of the characters in the string are used as a list of byte values.

If quotation marks are to be used within a string, they must be either the opposite set or they must be doubled.

Examples of the correct uses of quotation marks:

```
`This "string" is correct.`  
`This ` `string' ` is correct.`  
"This `string' is correct."  
"This " "string" " is correct."
```

Examples of incorrect uses of quotation marks:

```
`This `string' is not correct.`  
"This "string" is not correct."
```

value is a hexadecimal value that is a maximum of four characters.

DEBUG COMMAND REFERENCE

ASSEMBLE

A [*address*]

Assembles Macro Assembler statements directly into memory.

Parameters

address is the starting address for the instructions to be assembled in memory.

If you omit the address, ASSEMBLE begins at the next address following the last assembled program. If no ASSEMBLE statement was used previously, assembly starts at CS:0100. All numeric values are hexadecimal and can be entered as 1-4 characters.

Notes and Suggestions

- If a statement contains a syntax error, DEBUG displays:

↑Error

and redisplay the current assembly address.
- 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.
- Prefix mnemonics must be specified in front of the opcode to which they refer.

Examples

Near/Far

```
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 assembler automatically assembles short, near, or far jumps and calls, depending on byte displacement, to the destination address. These may be overridden with the NEAR or FAR prefix. You can abbreviate the NEAR prefix as NE.

Word Ptr/Byte Ptr

```
NEG          BYTE PTR [128]
DEC          WD [S1]
```

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 the prefix WORD PTR or BYTE PTR. These can be abbreviated WD and BY.

Memory Location/Immediate Operand

```
MOV          AX,21          ;Load AX with 21H
MOV          AX,[21]        ;Load AX with the contents of
                             ;memory location 21H
```

You use square brackets to tell DEBUG that an operand refers to a memory location rather than to an immediate operand. DEBUG uses the convention that operands enclosed in square brackets refer to memory.

DB/DW Opcodes

```
DB           1,2,3,4,"THIS AN EXAMPLE"
DB           'THIS IS A QUOTE:'
DB           "THIS IS A QUOTE:"
DW           1000,2000,3000, "BACH"
```

Two popular pseudo-instructions are available with the ASSEMBLE command. The DB opcode assembles byte values directly into memory. The DW opcode assembles word values directly into memory.

Register Indirect Commands

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

All forms of register indirect commands are supported.

Opcode Synonyms

```
LOOPZ    100
LOOPE    100
JA       100
JNBE     100
```

All opcode synonyms are also supported.

WAIT/FWAIT

```
FWAIT FADD ST,ST(3) ; This line will assemble
                        ; FWAIT prefix
FLD TBYTE PTR [BX]  ; This line will not
```

The WAIT or FWAIT must be explicitly specified for 8087 opcodes.

COMPARE

C range address

Compares memory specified by *range* to a memory block of the same size beginning at *address*.

Parameters

range is the beginning location and the number of bytes of the first block of memory to be compared.

address is the beginning address of the block of memory to which *range* is compared.

If the two areas of memory are different, DEBUG displays the differences in this format:

address1 byte1 byte2 address2

address1 byte1 refers to the address and contents of a location in the specified range.

byte2 address2 refers to the corresponding address and contents in the block starting at *address*.

Notes and Suggestions

- If the two areas of memory are identical, DEBUG only displays its prompt (—).
- If you enter only an offset for the starting address of *range*, the segment indicated by Register DS is used.

Examples

```
C 100,1FF 300   
C 100L100 300 
```

are two commands that have the same effect. Each command compares the block of memory from DS:100 to DS:1FF with the block of memory from DS:300 to DS:3FF.

DUMP

D [*address*]

D [*range*]

Displays the contents of the specified *address* or *range* in memory.

Parameters

address is the beginning memory location for DUMP.

range is the number of bytes to display.

Notes and Suggestions

- The dump is displayed in two portions:
 - A hexadecimal portion where each byte is displayed in hexadecimal.
 - An ASCII portion. Each byte is displayed as an ASCII character. Characters that cannot be displayed are shown as periods (.).
- Each displayed line begins on a 16-byte boundary and shows 16 bytes. A hyphen appears between Byte 8 and Byte 9.
- If you specify only an address with the DUMP command, DEBUG displays the contents of memory, starting at the address.
- If you specify neither parameter, DEBUG displays 128 bytes, beginning immediately after the last address displayed by a previous DUMP command.
- If you specify only an offset for the starting address, the segment indicated by register DS is used.

Example

```
D CS:100 109 
```

DEBUG displays the contents of the range C:100 109 in the following format:

```
04BA:0100 54 4F 4D 20 53 41 57 59-45 52 TOM  
SAWYER
```

```
D 
```

displays the dump as formatted in the preceding example. Each line of the display begins with an address, incremented by 16 from the address on the previous line. Each subsequent D (typed without parameters) displays the byte immediately following those last displayed.

```
D CS:100 L 20 
```

displays the dump as formatted in the first example, except this example displays 20 hex bytes.

```
D CS:100:115 
```

displays the dump as formatted in the first example, except this example displays all bytes in the range 100 to 115 hex.

ENTER

E *address* [*list*]

Enters byte values into memory at the specified address.

Parameters

address is the beginning location of the memory block to receive new values.

list is one or more new byte values to enter.

Notes and Suggestions

- If you enter only an offset for the address, the segment indicated by register DS is used.
- If you type the optional list of values, DEBUG replaces the contents of memory, beginning at *address*, with the new values. For example:

```
E DS:100 45 A1 "abc" 0F
```

places the six bytes in the list into memory beginning at DS:100.

If an error occurs, ENTER does not change the byte values.

- If you omit the list, DEBUG displays the address and its contents, and then waits for your input. You can do any of the following:
 - Enter a hexadecimal byte value to replace the displayed value. (Illegal or extra characters are ignored.)
 - Press the space bar to advance to the next byte. To change this value, type the new value as described above. Each press of the space bar advances to the next byte without changing the current byte.

If you space beyond an 8-byte boundary, DEBUG starts a new display line.

- Press the hyphen (-) to back up to the preceding byte. The preceding address and its contents are displayed on the next line. If you want to change this byte, type the new value as described above. Each press of backs up one more byte without changing the current byte.
- Press to end the ENTER command.

Examples

```
E CS:100 
```

causes DEBUG to enter byte values into memory beginning at CS:100. DEBUG displays the address and its contents (EB) as shown here.

```
04BA:0100 EB.
```

To change the value from EB to 41, type 41 (at the present cursor position) and press the space bar. DEBUG stores the value 41 and displays the contents of the next byte:

```
04BA:0100 EB.41 10.
```

To display the contents of the next two bytes, press the space bar twice more. You might see:

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

To change BC to 42, type 42 as shown:

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

If you want to go back and change 10 to 6F, press twice to return to value 10, and then type 6F:

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

Press to end the ENTER command and return to the DEBUG command level.

FILL

F range list

Fills the memory locations in the specified range with the values in the list.

Parameters

<i>range</i>	is the area of memory to receive the new values.
<i>list</i>	is a list of one or more data values to place in <i>range</i> .

Notes and Suggestions

- If *list* contains fewer bytes than the range, DEBUG repeatedly uses the values in the list until all locations in the range are filled.
- If *list* contains more bytes than the range, DEBUG ignores the extra values in the list.
- If you enter only an offset for the starting address of *range*, DEBUG uses the segment indicated by Register DS.

Example

```
F 04BA:100 L 100 42 45 52 54 41 ENTER
```

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

GO

G [= *address1*[*address2*...]]

Executes a program currently in memory. Execution stops at specified breakpoints, and the registers, flags, and instruction line for the next instruction to execute are displayed.

Parameters

address1 is the start location of the program to execute.
address2... are the addresses at which you want optional breakpoints.

Notes

- If you include *address1*, execution begins at *address1* in the CS segment. You must include the equal sign (=).
- If you omit *address1*, the program execution starts at the current instruction. DEBUG determines the address of the current instruction from the contents of the CS and IP registers.
- *address2* lets you set breakpoints. These allow you to examine the register contents and flag settings in effect when a specified address is reached during program execution.

Execution stops at the first breakpoint address encountered, regardless of that address's position in the list of breakpoint addresses. When the program execution reaches the breakpoint, DEBUG displays the registers, flags, and decoded instructions for the last instruction executed.

After execution halts at a breakpoint, you can enter the GO command again. The program resumes execution at the instruction after the breakpoint.

- If you enter the GO command without parameters, DEBUG executes the program as if you were executing the program outside DEBUG.
- An interrupt code (0CCH) is placed at the specified breakpoint address(es). When an instruction with the breakpoint code is reached, DEBUG restores all breakpoint addresses to their original instructions. If no breakpoint is reached, the original instructions are not restored.
- You can specify a maximum of 10 breakpoint addresses, in any order. If you enter only an offset for a breakpoint address, the segment indicated by register CS is used. You can set breakpoints only at addresses that contain the first byte of an opcode. Execution stops when any breakpoint is reached.
- The user stack pointer must be valid, and must have six bytes available for the GO command. This command uses an IRET instruction to cause a jump to the program under test. The user stack pointer is set and the user flags, CS 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 might crash.) An interrupt code (0CCH) is placed at the specified breakpoint address(es).

Example

G CS:7550 ENTER

causes the program currently in memory to execute up to address 7550 in the CS segment. DEBUG restores the original instructions, and displays the register contents and the flags. Then, the GO command ends.

HEX

H *value1 value2*

Performs hexadecimal arithmetic on *value1* and *value2*.

Parameters

value1 value2 DEBUG first adds *value1* and *value2* and then subtracts *value2* from *value1*. It displays the sum and the difference on one line.

Example

```
H 19F 10A 
```

causes DEBUG to perform the arithmetic, and displays the results:

```
02A9 0095
```

The sum of 19F and 10A is 02A9, and the difference is 0095.

INPUT

I *portaddress*

Inputs and displays one byte from the specified port.

Parameters

portaddress is a 16-bit port address in hexadecimal.

Example

```
I 2F8 ENTER
```

causes DEBUG to input the byte at Port 2F8 and display it. If the byte is 42 hex, DEBUG displays:

```
42
```

LOAD

L [*address*[*drive sector sectorcount*]]

Loads a file into memory, and sets BX:CX to the number of bytes read.

Parameters

<i>address</i>	is the beginning memory location for a file load.
<i>drive sector</i>	specifies the absolute disk sectors that contain the file to load.
<i>sectorcount</i>	is the number of disk sectors to load.

Notes and Suggestions

- Before you can load a file, you must have named it either when starting DEBUG or with the NAME command. Both procedures format a filespec properly in the File Control Block at CS:5C.
- If you omit all parameters, DEBUG loads the file into memory at address CS:100, and sets BX:CX to the number of bytes loaded. If you specify *address*, DEBUG loads the file beginning at the specified address in memory. In both cases, the file is read from the drive specified in the filespec or from the default drive if no drive was specified.
- If you include all parameters, DEBUG loads the absolute disk sectors. It loads them from the specified drive (0 = Drive A, 1 = Drive B, and so on). DEBUG begins loading with the specified sector and continues until the number of sectors indicated by *sectorcount* are loaded.
- When you use an offset for the starting address, DEBUG uses the segment indicated by register CS.

- When you load a file with an .exe extension, DEBUG relocates the file to the load address specified in the header of the .exe file, and ignores any address you may have specified. It strips the header from the .exe file before loading the file. Thus, the size of an .exe file on disk differs from its size in memory.
- If the file you indicated with the NAME command, or specified when you started DEBUG, is a .hex file, then entering the LOAD command without parameters causes DEBUG to load the file at the address specified in the .hex file. If the LOAD command includes the address option, DEBUG adds the specified address to the address found in the .hex file to determine the start address for loading the file.

Example

```
DEBUG   
N file.com   
L 
```

DEBUG loads the file called File.com from the default disk. To load only portions of a file or certain sectors from a disk, type a command similar to the following:

```
L 04BA:100 2 0F 6D 
```

DEBUG loads 6DH (109) consecutive sectors into memory from Drive C, beginning with absolute sector number 0F (15). The data is placed beginning at address 04BA:0100.

MOVE

M *range address*

Moves the block of memory specified by *range* to the location beginning at *address*.

Parameters

<i>range</i>	is the beginning location and the number of bytes of memory to move.
<i>address</i>	is the beginning address location to receive the data.

Notes and Suggestions

- Overlapping moves (where some locations in the range are also in the block beginning at *address*) are performed without loss of data. This is because MOVE transfers the addresses that it would otherwise overwrite before transferring the addresses that would overwrite those addresses.

When moving higher addresses to lower addresses, MOVE transfers the data beginning at the block's lowest address first, and then works toward the highest address. When moving from lower addresses to higher addresses, MOVE transfers the block's highest address first, and then works toward the lowest address.

If the operation does not write over the addresses in the block being moved, the data that resides there is unchanged. MOVE copies data from one area to another, in the sequence described, and writes over the new addresses.

- If you use only an offset for the starting address of *range* or for *address*, DEBUG uses the segment indicated by register DS. To specify an ending address for the range, enter only an offset value.

Example

```
M CS:100 110 CS:500 
```

DEBUG moves 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 review the results of MOVE, use DUMP with the same address used for MOVE.

NAME

N *filespec1* [*filespec2*...]

Assigns filespecs or program names for later LOAD or WRITE commands. If you enter DEBUG without naming any file, then you must specify a filename using the NAME command before a file can be loaded. NAME also assigns filespec parameters to the file being debugged.

Parameters

filespec1 filespec2 are names assigned to the files you use with DEBUG.

Notes

- The four areas of memory that NAME affects are:
 - DS:5C File Control Block for *file1*
 - DS:6C File Control Block for *file2*
 - DS:80 Count of characters
 - DS:81 All characters typed
- A File Control Block (FCB) for the first file-spec parameter given to the NAME command is set up at DS:5C. If you include *filespec2*, an FCB is set up for it at DS:6C. DS:80 contains the number of characters typed after the letter N in the NAME command line. All characters typed after the N are stored beginning at DS:81.

Examples

```
N file1.exe   
L   
N file2.dat file3.dat   
G 
```

The NAME command sets File1.exe as the filespec for the LOAD command that follows. After loading File1.exe into memory, use the NAME command to specify the parameters for File1.exe. When GO is executed, File1.exe is executed as if File1, File2.dat, and File3.dat are entered at the MS-DOS command level.

```
DEBUG prog.com   
N param1 param2/c   
G 
```

GO executes the file in memory as if Prog, *param1*, and *param2/C* are typed at the MS-DOS command level. Testing and debugging therefore reflect a normal runtime environment for Prog.com.

OUTPUT

O *portaddress byte*

Sends the byte to the specified portaddress.

Parameters

portaddress byte is a 16-bit port address.

Example

```
O 2F8 4F ENTER
```

causes DEBUG to output the byte value 4F to Port 2F8.

PROCEED

P [= *address*] [*value*]

Executes one or more instructions, and displays the register contents, flags, and the next instruction after each. PROCEED is similar to TRACE. The differences are that PROCEED:

- Executes automatically.
- Returns from any call or software interrupt instructions.
- Executes loop and repeat string instructions.

Parameters

address is the address of the instruction at which PROCEED is to begin.

value is the number of instructions to execute, beginning with *address*.

Examples

P

displays the registers and flags for the current instruction.

P=011A 10

executes 16 (10 hex) instructions, beginning at 011A in the current segment, and displays the registers and flags after each execution.

QUIT

Q

Ends the DEBUG program. The QUIT command exits DEBUG without saving the file you are debugging, and returns to the MS-DOS command level.

Parameters

None

Example

Q

ends DEBUG, and returns to the MS-DOS system prompt.

REGISTER

R [*registername*]

Performs any of these functions:

- Displays the contents of all registers and flag settings.
- Displays the contents of a single register, and lets you change the contents.
- Displays the flag settings, and lets you change the settings.

Parameters

registername is the name of the register displayed.

Notes and Suggestions

- If you enter **R** with no *registername*, **DEBUG** displays the contents of all registers and flags, with the next instruction to be executed.
- If you include a *registername*, **DEBUG** displays the 16-bit value of that register in hexadecimal, and then displays a colon prompt. Change the contents of the register by entering a 1- to 4-character hexadecimal value, or leave the contents unchanged by pressing **[ENTER]**. The valid *registernames* are:

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

Any other entry for the *registername* causes a **BR** error message.

Both **IP** and **PC** refer to the Instruction Pointer.

- Entering **F** as the *registername* causes **DEBUG** to display a 2-letter status code for each flag, showing whether it is set or clear. To change any flag, enter the opposite code.

- The flags are listed below with their codes for set and clear:

Flag Name	Set	Clear
Overflow (yes/no)	OV	NV
Direction (decrement/ increment)	DN	UP
Interrupt (enable/disable)	EI	DI
Sign (negative/positive)	NG	PL
Zero (yes/no)	ZR	NZ
Auxiliary carry (yes/no)	AC	NA
Parity (even/odd)	PE	PO
Carry (yes/no)	CY	NC

Typing **RF** causes **DEBUG** to display the flags in the order shown above at the beginning of a line. The hyphen prompt (-) appears at the end of the line. Now you can change any of the flag values by typing alphabetic pairs, in any order. You need not leave spaces between the flag entries. Press **ENTER** to make the changes and exit the command. Flags for which you did not type a new value remain unchanged.

If you make an error in your entry, **REGISTER** changes the flags up to the error in the list. It does not change the flags at or following the error.

At startup, **DEBUG**:

- Sets the segment registers to the bottom of free memory.
- Sets the instruction pointer to 0100H.
- Clears all flags.
- Sets the remaining registers to zero.

Examples

R

causes DEBUG to display all registers, flags, and the next instruction to be executed. For example, if the location is CS:11A, the display looks 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
```

RF

DEBUG displays the flags:

```
NV UP DI NG NZ AC PE NC-
```

Type one or more flag designations, in any order, with or without intervening spaces. For example:

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

DEBUG makes the requested changes.

R AX

DEBUG displays the contents of the single register AX:

```
AX 0E00
:_
```

To change the contents to 00FF, enter FF after the colon prompt.

SEARCH

S range list

Searches the specified area of memory for a list of bytes.

Parameters

range specifies the start and end addresses of the area to search.

list are the byte values to find.

Notes and Suggestions

- The list can contain one or more bytes, each separated by a space or comma. DEBUG displays the starting address for each match found.
- If the list contains more than one byte, SEARCH returns only the first address of the byte sequence. If the list contains only one byte, SEARCH returns all addresses of the byte in the specified range.
- If it cannot find the list, DEBUG doesn't display any address.
- If you enter only an offset for the starting address of range, the segment indicated by register DS is used.

Example

```
S CS:100 110 41 ENTER
```

causes DEBUG to search the addresses from CS:100 to CS:110 for 41H. If it finds two matches, DEBUG displays a response similar to this:

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

TRACE

T [= *address*][*value*]

Executes one or more instructions, and displays the register contents, flags, and next instruction after each.

Parameters

=address is the address of the instruction at which you want the trace to begin.

value is the number of instructions to trace (beginning with *address*).

Notes and Suggestions

- If you enter T with no parameters, DEBUG executes the instruction at CS:IP (the current instruction), and displays the registers and flags. If you enter the optional *=address*, tracing begins at the specified address. The optional *value* causes DEBUG to execute and trace the number of instructions specified by *value*.
- When tracing more than one instruction, you can suspend the display by pressing **CTRL** **S**. This lets you study the registers and flags for any instruction. Press the space bar to continue scrolling.
- The TRACE command uses the hardware trace made of the 8086 and 8088 microprocessors. Consequently, you can also trace instructions stored in ROM (read-only memory).

Examples

T

causes DEBUG to display the registers and flags for the current instruction. If the current instruction is 04BA:011A, DEBUG might display:

```
AX=0E00 BX=00FF CX=007 DX=0177
SP=039D BP=0000 SI=005C DI=0000
DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A NV UP DI NG NZ AC PO NC
04BA:011A CD21      INT      21
```

T=011A 10

DEBUG executes 16 (10 hex) instructions beginning at 011A in the current segment, and displays the registers and flags after each instruction.

UNASSEMBLE

U [*address*]

U [*range*]

Disassembles instructions and displays their addresses, their hexadecimal values, and the source statements that correspond to them.

Parameters

address is the address of the instructions to disassemble.

range is the number of bytes to disassemble.

Notes and Suggestions

- The display of disassembled code looks like a listing for an assembly language source file. The UNASSEMBLE command, without parameters, starts immediately following the end of the last disassembled memory. If you specify *address*, instructions are disassembled beginning with *address*.
- If you enter the UNASSEMBLE command with the range specified, DEBUG disassembles all bytes in the range.
- In all cases, DEBUG might disassemble and display slightly more bytes than the default amount or the number you requested. This is because instructions are of varying lengths, and the last instruction disassembled might include more bytes than expected.
- When you use only an offset for the starting address, the segment indicated by register CS is used.
- If you have altered some locations, using DEBUG, you can enter the UNASSEMBLE command for the changed locations, view the new instructions, and use the disassembled code to edit the source file.

Example

U04BA:100 L10

causes DEBUG to disassemble 16 bytes, beginning at address 04BA:0100, and to display information similar to the following:

-u 4061

2E40:4061	E89703	CALL	43FB
2E40:4064	B020	MOV	AL,20
2E40:4066	8AC8	MOV	CL,AL
2E40:4068	32C0	XOR	AL,AL
2E40:406A	A23F96	MOV	[963F],AL
2E40:406D	A05D96	MOV	AL,[965D]
2E40:4070	2402	AND	AL,02
2E40:4072	7503	JNZ	4077
2E40:4074	E8FDF9	CALL	3A74
2E40:4077	5E	POP	SI
2E40:4078	87DE	XCHG	SI,BX
2E40:407A	56	PUSH	SI
2E40:407B	51	PUSH	CX
2E40:407C	8D0E5896	LEA	CX,[9658]
2E40:4080	A05C96	MOV	AL,[965C]

WRITE

W [*address*[*drive sector sectorcount*]]

Writes the data being debugged to a disk file.

Parameters

<i>address</i>	is the memory location of the data saved.
<i>drive</i>	designates the disk drive to use for the save.
<i>sector</i>	designates the starting diskette sector for the save.
<i>sectorcount</i>	indicates the number of sectors to use in the save.

Notes and Suggestions

- If you omit all parameters, the file save begins at CS:100. If you specify only the address, the WRITE command writes the file beginning at that address. In either case, you must be certain that BX:CX contains the number of bytes to write. This value might have been set correctly by the DEBUG or LOAD commands, but later altered by a GO or TRACE command. (Note that if you load and modify a file, the name, length, and starting address are already set correctly as long as the length remains the same.)
- You must previously name the file either with the DEBUG startup command or with the NAME command. Both commands format a filespec properly in the File Control Block at CS:5C. DEBUG writes the file to the drive specified in the filespec or to the default drive if you don't specify the drive.

- If you specify all parameters, the write begins from the memory address specified. The file is written to the specified drive (0 = Drive A, 1 = Drive B, and so on). DEBUG writes the file beginning at the sector specified by *sector*, until the number of sectors specified by *sector-count* are written.
- If you use only an offset for the starting address, the segment indicated by register CS is used.
- If a disk write error occurs, DEBUG displays a message. Press **F3** to redisplay the WRITE command. Then, press **ENTER**.
- Be very careful when you write to absolute sectors on the disk. The write destroys existing data in these sectors.

Example

```
W CS:100 1 37 2B ENTER
```

writes the contents of memory to the disk in Drive B, beginning with memory address CS:100. The data is written beginning at relative sector 37H, and consists of 2BH sectors. The DEBUG prompt returns when the write is complete.

MESSAGES AND ERRORS

The MS-DOS operating system has a wide array of messages it displays. These messages serve various purposes, ranging from simply keeping you informed of what your computer is doing to warning you of difficulties your computer is having.

Some messages both provide information and request an action. For instance, the following error message requires you to press **A**, **R**, or **I** in response:

```
Disk error reading drive B
Abort, Retry, Ignore?
```

Other messages indicate an error, but do not necessarily require an action from you. For example, the following message states that you tried to copy a file to the same directory, using the same name:

```
File cannot be copied onto itself
0 files copied
```

The copy process automatically terminates after displaying the message.

Still other messages are for your information, and neither require action nor indicate errors. For instance, if two diskettes you compare with DISKCOPY are identical, DISKCOPY displays:

```
Compare OK
```

This chapter provides an alphabetical list of the MS-DOS messages. Keep in mind that an application program you are running might generate messages of its own. So, if an error message appears while you are running an application, and you cannot find the message here, check your application manual.

The Messages

(.) (..) does not exist
[CHKDSK]

This message indicates that either the current or parent directory is invalid.

Abort edit (Y/N)?

[EDLIN]

MS-DOS displays this message when you choose the Q (QUIT) command in EDLIN. QUIT exits the editing session **without** saving any editing changes. Press ☐ Y (for yes) to quit the edit or ☐ N (for no) to continue the edit.

Abort, Retry, Ignore?

[MS-DOS]

If a disk or device error occurs at any time during execution of a command or program, MS-DOS displays this message, asking you to terminate the command or program, retry it, or ignore the error.

Access denied

[REPLACE] [XCOPY]

You tried to replace or copy to a write-protected, read-only, or locked file.

All files cancelled by operator

[PRINT]

MS-DOS displays this message when you specify the /T switch with the PRINT command.

Allocation error in file, size adjusted

[CHKDSK]

The size of the file indicated in the directory is not consistent with the amount of data actually allocated to the file.

Allocation error, size adjusted

[CHKDSK]

The File Allocation Table (FAT) of the specified filename contains an invalid sector number. CHKDSK automatically truncates the file to the end of the last valid sector.

All specified files are contiguous

[CHKDSK]

All files are allocated contiguously on the disk without fragmentation.

Are you sure (Y/N)?

[MS-DOS]

MS-DOS displays this message if you try to delete all files in a directory. Press **[Y]** (for yes) if you want to delete the files. Otherwise, press **[N]** (for no).

Attempted write-protect violation

[FORMAT]

The disk you are trying to format is write-protected.

**Attempt to access data outside of segment bounds,
possibly bad object module**

[LINK]

This usually indicates that a bad object file exists.

Bad call format reading drive (x:)

[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

Bad call format writing drive (x:)

[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

Bad command error reading drive (x:)

[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

Bad command error writing drive (x:)

[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

Bad command or file name

[MS-DOS]

The command cannot find the specified program. Check the spelling of the filename, and be sure the file exists on the disk.

Bad file

[FC]

A file you specified is defective.

Bad numeric parameter

[LINK]

A numeric value is not given as digits.

Bad or missing Command Interpreter

[MS-DOS]

MS-DOS cannot find the COMMAND.COM file on the disk. Either the file is missing from the ROOT directory, or the file is invalid. Restart the system, or copy the COMMAND.COM file from your master MS-DOS system disk to your working system disk. You also receive this message if COMMAND.COM is not in the same directory it was in when you started MS-DOS.

Bad or missing (*filename*)

[MS-DOS]

You specified an invalid device in the CONFIG.SYS file. Check the accuracy of the DEVICE command in CONFIG.SYS.

Bad Partition Table

[FORMAT]

There is no DOS partition on the hard disk. Use FDISK to create a DOS partition so you can use the disk.

Bad switch syntax

[MS-DOS]

You used an invalid switch syntax with a command. Check the command reference section for valid switches.

Bad unit error reading drive (x):

[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

BF Error (Bad flag)

[DEBUG]

The characters entered to alter *flag* are not valid flag values. See the explanation of the REGISTER command for a list of valid flag entries.

BP Error (Too many breakpoints)

[DEBUG]

You included more than 10 breakpoints in the GO command. Retype the GO command with 10 or fewer breakpoints.

Break is off (or on)

[MS-DOS]

This message tells you the current setting of BREAK.

BR Error (Bad Register)

[DEBUG]

You gave the REGISTER command an invalid register name. See the REGISTER command for a list of valid names.

**Cannot CHDIR to *filename*-
tree past this point not processed**

[CHKDSK]

CHKDSK is traveling the tree structure of the directory, and is unable to go to the specified directory. All subdirectories underneath this directory are not verified.

**Cannot CHDIR to root
Processing cannot continue**

[CHDIR]

CHKDSK is traveling the tree structure of the directory, and is unable to return to the ROOT directory. CHKDSK is not able to continue checking the remaining subdirectories of the ROOT.

Cannot CHKDSK a Network drive

[CHKDSK]

You cannot check drives that are redirected over the network.

Cannot CHKDSK a SUBSTed or ASSIGNED drive

[CHKDSK]

You cannot check drives that have been substituted or assigned.

**Cannot DISKCOMP to or from
a network drive**

[DISKCOMP]

You cannot compare a disk on drives that are redirected over the network.

**Cannot DISKCOMP to or from
an ASSIGNED or SUBSTed drive**

[DISKCOMP]

One of the drives that you specified is a drive created using the ASSIGN or SUBST command.

**Cannot DISKCOPY to or from
a network drive
[CHKDSK]**

You cannot copy disks to or from drives that are redirected over the network.

**Cannot do binary reads from a device
[COPY]**

You cannot do a copy in the binary mode when you are copying from a device. Remove the /B switch, or specify an ASCII copy, using the /A switch.

**Cannot edit .BAK file—rename file
[EDLIN]**

You cannot edit a backup copy created by EDLIN. Either rename the file, or copy the .bak file, giving it a different extension.

**Cannot format an ASSIGNED or SUBSTed drive
[FORMAT]**

You cannot format a drive that is mapped to another drive using the ASSIGN or SUBST command. Run the FORMAT command again after clearing all drive assignments.

**Cannot FORMAT a Network drive
[FORMAT]**

You cannot format drives that are redirected over the network.

**Cannot label a network drive
[LABEL]**

You cannot label a drive that is shared on a network server station.

**Cannot LABEL a SUBSTed or ASSIGNED drive
[LABEL]**

You cannot label a drive that has been substituted (using SUBST) or assigned (using ASSIGN).

**Cannot open (*filename*)
[PRINT]**

Either MS-DOS cannot find the specified file to print, or the file does not exist. Check the command for a valid filename.

Cannot open temporary file
[LINK]

The linker cannot create the file VM.TMP because the disk directory is full. Insert a new disk. Do not remove the disk that is to receive the listing (map) file.

Cannot perform a cyclic copy
[XCOPY]

When using the /S switch, you cannot specify a target that is a subdirectory of the source.

Cannot recover . entry, processing continued
[CHKDSK]

The . entry (current directory) is defective.

Cannot recover .. entry, processing continued
[CHKDSK]

The .. entry (parent directory) is defective.

Cannot Recover a Network drive
[RECOVER]

You cannot recover files on drives redirected over the network.

Cannot SUBST a network drive
[SUBST]

You cannot substitute drives that are redirected over the network.

Cannot SYS to a Network drive
[SYS]

You cannot transfer the system file to drives that are redirected over the network.

Cannot XCOPY from a reserved device
[XCOPY]

You cannot copy from a device to a file.

Cannot XCOPY to a reserved device
[XCOPY]

You cannot copy files to a device.

**CHDIR .. failed, trying alternate method
[CHKDSK]**

In traveling the tree structure, CHKDSK cannot return to the parent directory. It tries to return to that directory by starting over at the ROOT and traveling down.

**Compare another diskette (Y/N)?
[DISKCOMP]**

DISKCOMP displays this message when it finishes comparing diskettes. Press ☐ Y (for yes) if you want to compare another pair of diskettes. Otherwise, press ☐ N (for no).

**Compare error on
side *s*, track *t*
[DISKCOMP]**

Your source and target diskettes are not identical. DISKCOMP found that the data on the side indicated by *s* and the track indicated by *t* does not match between the two diskettes.

**Compare OK
[DISKCOMP]**

The compared diskettes are identical.

**Compare process ended
[DISKCOMP]**

A fatal error occurred during the comparison.

**Comparing (*tt*) tracks
(*xx*) sectors per track, (*y*) side(s)
[DISKCOMP]**

This message confirms the format of the disks you are comparing.

**Content of destination lost before copy
[MS-DOS]**

A file to be used as a source file in the COPY command was overwritten prior to completion of the copy. The following is an example of a command that gives such an error:

```
COPY file1+file2 file2 
```


Convert lost chains to files (Y/N)?

[CHKDSK]

If you press ☐ Y (for yes) in response to this prompt, CHKDSK recovers the lost blocks it found when checking the disk. CHKDSK creates a directory entry and a file for you with the filename FILEnnnn.CHK. If you press ☐ N (for no) in response to this prompt, CHKDSK frees the lost blocks so they can be reallocated.

Copy complete

Copy another (Y/N)?

[DISKCOPY]

DISKCOPY is finished copying. Press ☐ Y (for yes) if you want to copy another disk. Otherwise, press ☐ N (for no).

Copy not completed

DISKCOPY

DISKCOPY is unable to copy the entire disk.

Copying...

[DISKCOPY]

This message indicates that DISKCOPY is copying a disk.

Corrections will not be written to disk

[CHKDSK]

There are errors on the disk, but you did not specify the /F switch to correct them.

Current date is (mm-dd-yy)

[MS-DOS]

MS-DOS displays this message in response to the DATE command.

Current time is (hh:mm:ss.cc)

[MS-DOS]

MS-DOS displays this message in response to the TIME command.

Data error reading drive (x)

[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

Delete current volume label (Y/N)?

[LABEL]

If a current volume label exists, LABEL displays this message when you press **[ENTER]** in response to the prompt to enter a new volume label. If you want to delete the label, press **[Y]** (for yes). Otherwise, press **[N]** (for no).

Destination disk format error

[FORMAT]

An error occurred during formatting. The disk is flawed, or you inserted it into the drive incorrectly.

DEVICE Support Not Present

[DISKCOMP]

The disk drive does not support MS-DOS 3.2 device control.

DF Error (Double flag)

[DEBUG]

You entered two values for one flag when using the REGISTER command. You can specify a flag value only once per RF command.

Directory is joined

[CHKDSK]

CHKDSK cannot process directories that are joined.

Directory is totally empty,

no . or ..

[CHKDSK]

The specified directory does not contain references to current and parent directories. Delete the specified directory, and recreate it.

Directory not empty

[JOIN]

You can join with a directory only if the directory is empty.

Disk error reading drive (x:)

[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

Disk error reading FAT
[CHKDSK]

One of your file allocation tables has a defective sector in it. MS-DOS automatically uses the other FAT. To ensure that you do not lose data, copy all your files onto another disk.

Disk error writing drive (x:)
[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

Disk error writing FAT
[CHKDSK]

Use the COPY command to copy all files to another disk.

Diskettes compare OK
[DISKCOMP]

The diskettes that you are comparing are identical.

Disk full. Edits lost
[EDLIN]

Because of a lack of sufficient disk space, EDLIN was not able to save your file.

Disk full—write not completed
[EDLIN]

You gave an END command when the disk did not contain enough free space for the file. EDLIN terminated and returned you to the operating system. Part of the file might have been written to the disk and saved. Delete the saved portion, and restart the editing session. The file is not available after this error.

Disks do not compare
[DISKCOPY]

DISKCOPY found a discrepancy between the disks. Perform the duplication again.

Disks must be the same size
[DISKCOPY]

Use COPY to copy files between the disks.

**Disk unsuitable for system disk
[FORMAT]**

The FORMAT program detected a bad track on the disk where the system files should reside. Use this disk only for storing data.

**Divide error
[ATTRIB]**

The processor has set the divide overflow flag. This is usually caused by a program attempting to divide by zero.

**Divide overflow
[MS-DOS]**

The processor has set the divide overflow flag. This is usually caused by a program attempting to divide by zero.

**Does name specify a file name
or directory name on the target
(F = file D = directory)?
[XCOPY]**

The target directory does not exist.

**Do not specify filename(s)
Command format: DISKCOMP d: d:[/1]/[8]
[DISKCOMP]**

You specified an incorrect switch or gave a filename in addition to a drive name.

**Drive letter must be specified
[FORMAT]**

You must specify the drive you want to format.

**Drive not ready
[PRINT]**

When PRINT attempted a disk access, the drive was not ready. PRINT keeps trying until the drive is ready.

**Drive types or diskette types
not compatible
[DISKCOMP]**

You cannot compare a single-sided diskette with a double-sided diskette or a high-density diskette with a standard diskette. Use FC to compare the files on such diskettes.

**Duplicate file name or File not found
[CHKDSK] [MS-DOS]**

You tried to rename a file to a filename that already exists, or the name you specified cannot be found.

**ECHO is off (or on)
[MS-DOS]**

This message tells you the current status of ECHO.

**End of input file
[EDLIN]**

The entire file was read into memory. If the file is read in sections, this message indicates that the last section of the file is in memory.

**Enter current Volume Label for drive (x):
[FORMAT]**

FORMAT asks you to enter the current volume label for verification before it formats the specified hard disk.

**Enter new date:
[MS-DOS]**

You must respond to this prompt when you start MS-DOS. Press or enter the date in the displayed format.

**Enter new time:
[MS-DOS]**

You must respond to this prompt when you start MS-DOS. Press or enter the time in the displayed format.

**Entry error
[EDLIN]**

The last command typed contained a syntax error. Retype the command, using the correct syntax, and press .

**Entry has a bad attribute (or link or size)
[CHKDSK]**

This message might be preceded by one or two periods that indicate which subdirectory is invalid. If you specified the /F switch, CHKDSK tries to correct the error.

**Error: dup record too complex
[LINK]**

A DUP record in an assembly-language module is too complex. Simplify the DUP record.

**Error: fixup offset exceeds field width
[LINK]**

An assembly-language instruction refers to an address with a short instruction instead of a long instruction. Edit the assembly-language source and reassemble.

**Error in .EXE file
[MS-DOS]**

The .exe file you want to load has an invalid internal format.

**Error reading/writing partition table
[FORMAT]**

FORMAT cannot read or write the partition table. Run FDISK on the disk. Then, try reformatting.

**Errors found, F parameter not specified
Corrections will not be written to disk
[CHKDSK]**

CHKDSK found errors on the disk. If you did not specify the /F switch, CHKDSK continues printing messages, but it does not correct the errors.

**Errors on list device indicate that it
may be off-line. Please check it.
[PRINT]**

Your printer is not turned on.

**Errors writing to the system cannot continue
[FORMAT]**

Be sure you have a diskette in the drive specified by FORMAT and that the drive latch is closed. Be sure the diskette is not write-protected. If the error persists, your diskette is unusable.

**Errors writing to the system sectors,
cannot continue
[FORMAT]**

The system sectors (boot, file allocation table, and directory) are required for the disk to be useful.

**Error writing boot sector to destination
[FORMAT]**

All floppy diskettes—both data and system diskettes—must have a boot sector so they can be used by the system. Use another diskette as the source diskette.

**Error writing to device
[MS-DOS]**

You tried to send too much data to a device. MS-DOS is unable to write the data to the specified device.

**EXEC failure
[MS-DOS]**

MS-DOS either found an error when reading a command, or the FILES command in the CONFIG.SYS file has the number of files set too low. Increase the value and restart MS-DOS.

**FCB unavailable reading drive (x):
[MS-DOS]**

See “Disk and Device Errors,” later in this chapter.

**FCB unavailable writing drive (x):
[MS-DOS]**

See “Disk and Device Errors,” later in this chapter.

**fc: cannot open (*filename*) - No such file or directory
[FC]**

One of the specified files does not exist. Check the directory for the correct filename.

fc: (filename1) longer than (filename2)
[FC]

FC has reached the end of one of the files in a file comparison, but the other file still has uncomparing data left.

fc: incompatible switches
[FC]

The switches you specified are not compatible (for example, /B and /L). Do not combine binary and ASCII comparison switches.

fc: out of memory
[FC]

You do not have enough memory to perform the comparison.

fc: no differences encountered
[FC]

The files are the same.

File allocation table bad
[MS-DOS]

The disk might be defective. Run CHKDSK with the /F switch to fix the disk.

File allocation table bad drive (x:)
[CHKDSK] [PRINT]

The disk might be defective. Run CHKDSK with the /F switch to fix the disk.

File cannot be converted
[EXE2BIN]

The input file is not in the correct format.

File cannot be copied onto itself
[COPY] [REPLACE] [XCOPY]

The source pathname you specified is the same as the target pathname.

File creation error
[MS-DOS] [XCOPY]

You are trying to add a new filename or to replace a file that already exists in the directory. If the file already exists, it is a read-only file, and you cannot replace it. Run CHKDSK on the disk to determine the cause of the error.

File is READ-ONLY
[EDLIN]

You cannot change the file because it is designated read-only.

(filename) cancelled by operator
[PRINT]

This message is printed on the printer when you specify the /T switch in the PRINT command.

(filename) contains non-contiguous blocks
[CHKDSK]

The specified file is not allocated contiguously on the disk. If you specify the /F switch, CHKDSK fixes this error.

filename is cross linked on cluster
[CHKDSK]

Make a copy of the file you want to keep. Then, delete the cross-linked files.

(filename) is currently being printed
[PRINT]

The specified file is printing.

(filename) is in queue
[PRINT]

The specified file is waiting to be printed.

Filename must be specified
[EDLIN]

You must specify a filename when you start EDLIN.

File not found
[EDLIN] [MS-DOS] [PRINT] [RECOVER]

MS-DOS cannot find the file you specified. Check to see that the pathname is accurate.

Files are different

[FC]

FC compares the portion of the files that it can load into the buffer space. If no lines in those portions match, FC displays this message.

FIND: Access denied

[FIND]

You cannot access the file. Be sure the disk is not write-protected.

FIND: File not found

[FIND]

The file you specified does not exist. Be sure you type the filename correctly.

FIND: Invalid number of parameters

[FIND]

You are specifying too many or too few options in the command line.

FIND: Invalid Parameter

[FIND]

One of the switches you are specifying is incorrect.

FIND: Read error in (*filename*)

[FIND]

The program cannot read the specified file.

FIND: Syntax error

[FIND]

Be sure you type the command correctly.

First cluster number is invalid, entry truncated

[CHKDSK]

The file directory entry contains an invalid pointer to the data area. If you specified the /F switch, CHKDSK truncates the file to a zero-length file.

FIRST diskette bad or incompatible
[DISKCOMP]

DISKCOMP cannot recognize the format of the source diskette. Run CHKDSK to help identify the problem.

Fixups needed - base segment (hex:)
[EXE2BIN]

The source (.exe) file contains information indicating that the load segment is required for the file. Specify the absolute segment address where the finished module is to be located.

For cannot be nested
[MS-DOS]

Nesting of FOR command is not allowed in a batch file.

Format another (Y/N)?
[FORMAT]

Press ☐ Y (for yes) to format another disk. Otherwise, press ☐ N (for no). If you accidentally press ☐ Y, you can cancel the formatting process by pressing ☐ CTRL ☐ C in response to the **Strike any key to begin formatting** prompt.

Format complete
[FORMAT]

FORMAT is finished formatting the disk.

Format failure
[FORMAT]

This message is usually displayed with an explanation of why MS-DOS cannot format the disk.

Format not supported on drive (x:)
[FORMAT]

You cannot use FORMAT to format this drive.

General failure reading drive (x:)
[MS-DOS]

See "Disk and Device Errors," later in this chapter.

General failure writing drive (x:)
[MS-DOS]

See "Disk and Device Errors," later in this chapter.

Graphics characters loaded

[GRAFTABL]

The GRAFTABL command displays this message after loading the table of graphics characters into memory.

Graphics characters already loaded

[GRAFTABL]

You already loaded the table of graphics characters into memory.

Has invalid cluster, file truncated

Incompatible system size

[CHKDSK] [SYS]

The system files occupy more space on the source disk than is available on the destination disk.

Head: (hh) Cylinder: (cc)

[FORMAT]

FORMAT displays the head and cylinder number of the track currently being formatted.

Incompatible system size

[SYS]

The system files occupy more space on the source disk than is available on the target disk.

Incorrect DOS version

**[ATTRIB] [CHKDSK] [DISKCOMP] [DISKCOPY] [EDLIN]
[FMAT2000] [FORMAT] [GRAFTABL] [MORE] [PRINT]
[RECOVER] [SHARE] [SYS] [TREE] [XCOPY]**

Many Version 2.x and 3.x utilities cannot run on earlier versions of MS-DOS. Some utilities run only under the exact version of MS-DOS for which they were configured.

Incorrect number of parameters

[JOIN] [SUBST]

You specified too many or too few options in the command line.

Incorrect parameter

[ASSIGN] [SHARE]

One of the options you are using is incorrect.

Input file read error

[LINK]

This usually indicates that a bad object file exists.

**Insert destination disk in drive (x:)
and strike any key when ready**

[SYS]

This message appears when you are using SYS to transfer the operating system with a single disk drive. Insert the appropriate disk in the drive, and press the space bar to begin processing.

**Insert diskette for drive (x:)
and strike any key when ready**

[MS-DOS]

This message appears when MS-DOS is copying files. Insert a disk in the appropriate drive, and press the space bar to begin processing.

**Insert diskette with batch file
and press any key when ready**

[MS-DOS]

The disk containing the batch file you specified is not in the drive you originally specified. Reinsert the disk that contains the batch file.

**Insert DOS disk in drive (x:)
and strike ENTER when ready**

[FORMAT]

You are specifying FORMAT with /S , but the disk in the current drive does not contain the MS-DOS system files. Insert a system disk in the current drive, and press **ENTER**.

Insert FIRST diskette in drive (x:)

[DISKCOMP]

DISKCOMP displays this message to prompt you to insert the first diskette in the specified drive.

**Insert new diskette for drive (x):
and strike ENTER when ready
[FORMAT]**

This message appears when you are using a single drive to format a diskette. Insert the diskette to be formatted in the appropriate drive, and press **[ENTER]** to begin formatting. If any data exists on the diskette, **FORMAT** writes over the data.

**Insert SECOND diskette in drive (x):
[DISKCOMP]**

DISKCOMP displays this message to prompt you to insert the second diskette you are comparing.

**Insert source diskette into drive (x):
[DISKCOPY]**

Insert the diskette to be copied into the specified drive.

**Insert system diskette in drive (x):
and strike any key when ready
[SYS]**

SYS needs a diskette from which to read the system files. Insert in the specified drive a diskette containing those files. Then, press the space bar to start the system copy process.

**Insert target diskette into drive (x):
[DISKCOPY]**

You are running **DISKCOPY**, and your source and target drives are the same. Reinsert the target diskette into the specified drive.

**Insufficient disk space
[MS-DOS] [REPLACE]**

The disk is full. It does not contain enough room to perform the specified operation.

**Insufficient memory
[CHKDSK] [DISKCOMP] [EDLIN] [REPLACE]**

There is not enough memory to perform the specified operation.

Insufficient memory for system transfer

[FORMAT]

Your memory configuration is insufficient to transfer the MS-DOS system file with the /S switch.

Insufficient room in root directory.

Erase files in root and repeat CHKDSK

[CHKDSK]

CHKDSK always recovers lost files into the ROOT directory. In this case, your ROOT directory is full. Delete some files to make room for the lost files.

Intermediate file error during pipe

[MS-DOS]

The pipe operation makes use of temporary files on the disk that MS-DOS automatically deletes after the piping operation is complete. An error occurred in one of these files.

Internal error

[FC]

This error indicates an internal logic error in the FC utility.

Invalid characters in volume label

[FORMAT] [LABEL]

The volume label can contain a maximum of 11 alphanumeric characters.

Invalid COMMAND.COM

**Insert COMMAND.COM disk in default drive
and strike any key when ready**

[MS-DOS]

The program you just ran uses almost all of memory. MS-DOS must now reload the COMMAND.COM file from disk. However, MS-DOS cannot find COMMAND.COM on the disk, or the copy found is invalid. Insert a disk that contains a copy of COMMAND.COM similar to the version that is on the disk you used to start MS-DOS.

Invalid country code

[MS-DOS]

The country number you specified in CONFIG.SYS is not valid.

**Invalid current directory
Processing cannot continue
[CHKDSK]**

Restart the system and rerun CHKDSK.

**Invalid date
[BACKUP] [DATE] [RESTORE] [XCOPY]**

Enter the date again, using the correct syntax.

**Invalid device
[MS-DOS]**

Valid devices are CON, NUL, AUX, and PRN.

**Invalid device parameters from device driver
[FORMAT]**

FORMAT displays this message when the number of hidden sectors is not evenly divisible by the number of sectors per track (the partition does not start on a track boundary). This might happen if you try to format a hard disk previously formatted with MS-DOS Version 2.x without first running FDISK.

**Invalid directory
[MS-DOS]**

The directory you specified either does not exist or is invalid. Be sure you enter the name correctly.

**Invalid disk change reading drive (x:)
[MS-DOS]**

See “Disk and Device Errors,” later in this chapter.

**Invalid disk change writing drive (x:)
[MS-DOS]**

See “Disk and Device Errors,” later in this chapter.

**Invalid drive in search path
[MS-DOS]**

The drive does not exist.

**Invalid drive or file name
[EDLIN] [RECOVER]**

Specify a valid drive name or filename.

Invalid drive specification

**[CHKDSK] [DISKCOMP] [DISKCOPY] [FORMAT] [PRINT]
[REPLACE] [SYS] [TREE] [XCOPY]**

You must specify a valid drive.

Invalid environment size specified

[COMMAND]

You are specifying an invalid number of bytes with the /E switch. Specify a number in the range 128 to 32768.

Invalid number of parameters

[MS-DOS] [ATTRIB] [RECOVER] [XCOPY]

You have specified an incorrect number of options in the command line.

Invalid object module

[LINK]

One or more object modules are incorrectly formed or are incomplete (as when assembly is stopped in mid-process).

Invalid parameter

**[CHKDSK] [DISKCOMP] [DISKCOPY] [EDLIN] [FORMAT]
[PRINT] [REPLACE] [SORT] [SUBST] [SYS] [TREE]
[XCOPY]**

A switch you specified is incorrect.

Invalid path or file name

[COPY] [XCOPY]

Specify a valid pathname or filename for this command.

Invalid path, not directory, or directory not empty

[MS-DOS]

You are unable to remove the directory requested for one of the reasons specified in the error message.

Invalid sub-directory entry

[CHKDSK]

The subdirectory you specified either does not exist or is invalid. Be sure you type the name correctly.

Invalid time

[BACKUP] [RESTORE] [TIME] [XCOPY]

Specify a valid time.

Invalid Volume ID

[FORMAT]

FORMAT displays this message if you enter a volume label that does not match the label on the hard disk you want to format. It then quits for format process.

Invalid working directory

[MS-DOS]

Your disk is bad. Replace the disk or make another copy from your backup system disk.

Label not found

[MS-DOS]

Your batch file contains a GOTO command that refers to a non-existent label.

Line too long

[EDLIN]

During a REPLACE command, the string given as the replacement caused the line to expand beyond 253 characters. Divide the long line into two lines, and retry the REPLACE command.

List output is not assigned to a device

[PRINT]

When you first run PRINT, it asks you for the device you want to specify as the print spooler. This message appears if PRINT is set up for a device that does not exist.

Lock violation reading drive (x:)

[MS-DOS]

See "Disk and Device Errors," later in this chapter.

Lock violation writing drive (x:)

[MS-DOS]

See "Disk and Device Errors," later in this chapter.

Memory allocation error. Cannot load MS-DOS, system halted

[MS-DOS]

Restart MS-DOS. If the error persists, use a new copy of the MS-DOS system diskette.

—More—

[MORE]

Press the space bar to view more of the file or directory.

MORE: Incorrect DOS Version

[MORE]

MORE does not run on versions of MS-DOS prior to 2.0.

Must specify destination line number

[EDLIN]

You must specify a destination line number when you are copying and inserting lines with EDLIN.

Must specify ON or OFF

[MS-DOS]

The command requires either an ON or an OFF argument.

Name of list device [PRN]:

[PRINT]

This prompt appears the first time you run PRINT. Specify any valid device as your PRINT output device.

New file

[EDLIN]

EDLIN displays this message if it cannot find a file that has the filename you specified. If you are creating a file, ignore this message. If you do not intend to create a file, check to see if you correctly typed the filename of the file you wish to edit.

No files added/replaced

[REPLACE]

The REPLACE command did not add or replace any files.

No files found (*filename*)

[REPLACE]

REPLACE cannot find source or target files that match the specified filenames.

No files match *d:xxxxxxxx.xxx*

[PRINT]

PRINT cannot find a file to match filename that you want added to the print queue.

No free file handles.

Cannot start COMMAND.COM, exiting

[MS-DOS]

Restart MS-DOS. If the message persists, increase the value given for FILES in the CONFIG.SYS file.

Non-DOS disk error reading drive (*x*)

[MS-DOS] [PRINT]

See “Disk and Device Errors,” later in this chapter.

Non-system disk or disk error

Replace and strike any key when ready

[FORMAT] [SYS]

Replace the disk with the proper disk, and press the space bar to continue.

No paper error writing device (*dev*)

[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

No path

[MS-DOS]

You are using the PATH command to determine the current search path for commands. There is no search path.

No room for system on destination disk

[SYS]

There is not enough room for system files on the target disk. Delete some files to make room, or use another disk. You might need to reformat the disk to put the system on it.

No room in directory for file

[EDLIN]

You are trying to save a file in the ROOT directory, but the directory is full. Use a subdirectory. Subdirectories are not limited in size as is the ROOT directory.

No subdirectories exist

[TREE]

The disk in the drive you specified does not contain subdirectories.

No system on default drive

[SYS]

The system files do not exist on the current disk.

Not enough memory

[JOIN] [SHARE] [SUBST]

There is not enough memory for MS-DOS to run the command.

Not enough room to merge the entire file

[EDLIN]

There was not enough room in memory to hold the file during a TRANSFER command. You must free some memory by writing some files to disk or by deleting some files before you can transfer this file.

Not found

[EDLIN]

You have specified a SEARCH or a REPLACE command that was unable to find a further occurrence of the specified SEARCH or REPLACE string.

Not ready error reading drive (x:)

[MS-DOS]

See "Disk and Device Errors," later in this chapter.

Not ready error writing drive (x:)

[MS-DOS]

See "Disk and Device Errors," later in this chapter.

O.K.?

[EDLIN]

This prompt occurs during SEARCH and REPLACE command processing. If you press any key except ☐Y or ☐ENTER, the SEARCH or REPLACE process continues.

Out of environment space

[MS-DOS]

There is not enough room in the program environment to accept more data. You can use COMMAND with the /E switch to increase the size of the environment.

Parameters not compatible

[FORMAT] [REPLACE]

You specified switches that you cannot use together.

Parameters not compatible with fixed disk

[FORMAT]

You are using a switch that is not compatible with the specified hard disk drive.

Parameters not supported

[MS-DOS] [FORMAT]

MS-DOS does not support a parameter you specified.

Parameters not supported by drive

[FORMAT]

FORMAT displays this message when the device driver for the specified drive does not support Generic IOCTL function requests.

Path not found

[CHKDSK] [REPLACE] [SUBST]

You specified an invalid path.

Path too long

[REPLACE] [XCOPY]

The pathname you specified is too long. You might have to change to a lower subdirectory to replace files in deep subdirectories.

Press any key to begin adding files
[REPLACE]

When you specify the /W switch, REPLACE displays this message to prompt you to start replacing files.

Press any key to begin recovery of the file(s) on drive (x):
[RECOVER]

This prompt appears before you recover a disk or file. Press the space bar to begin the recovery. If you wish to cancel the command, press **CTRL** **C**.

Press any key when ready
[DISKCOMP] [DISKCOPY]

This prompt appears when you enter a command to copy a disk. After inserting the diskettes into the appropriate drives, press the space bar to begin the copy. If you want to end the command, press **CTRL** **C**.

PRINT queue is empty
[PRINT]

There are no files waiting to be printed.

PRINT queue is full
[PRINT]

You can place a maximum of 10 files at a time in the print queue.

Probable non-DOS disk
Continue (Y/N)?
[CHKDSK]

The disk you are using is not recognized by this version of MS-DOS. The disk either was created by another system that is not supported on this version of MS-DOS, or it is not an MS-DOS disk. Do not continue processing if CHKDSK returns this message for a floppy diskette. If the message appears for a hard disk, the information describing the characteristics of the disk to MS-DOS is destroyed. In this case, you can continue CHKDSK processing by pressing **Y**.

Processing cannot continue
[CHKDSK]

There is not enough memory in your computer to process CHKDSK for the specified disk. You must obtain more memory to run CHKDSK.

Program size or number of segments exceeds capacity of linker
[LINK]

The total size cannot exceed 384K bytes, and the number of segments cannot exceed 255.

Program too big to fit in memory
[MS-DOS]

You must acquire more memory to run your application. It is possible that some programs you have run are still using some memory. Try to restart MS-DOS. If you still receive the message, you must acquire more memory.

Read error in (*filename*)
[FC] [FIND]

MS-DOS cannot read the file.

Read fault error reading drive (x:)
[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

Reading source file(s)...
[XCOPY]

XCOPY is now reading the source files you specified.

Re-insert diskette for drive (x:)
[FORMAT]

Reinsert the disk being formatted in the indicated drive.

Requested stack size exceed 64K
[LINK]

Specify a size less than or equal to 64K bytes with the /STACK switch.

Resident part of PRINT installed
[PRINT]

This is the first message MS-DOS displays when you issue the PRINT command. It means available memory has been reduced by several thousand bytes to process the PRINT command concurrently with other other processes.

Resynch failed. Files are too different.
[FC]

FC displays this message if the number of lines in the internal buffer is less than the number of consecutive, differing lines. Specify the /LB switch with a larger number if you want to display all the file differences.

SECOND diskette bad or incompatible
[DISKCOMP]

The second diskette does not contain the same format as the first, or DISKCOMP does not recognize the format of the second diskette. Run CHKDSK to help identify the problem.

Sector not found error reading drive (x:)
[MS-DOS]

See "Disk and Device Errors," later in this chapter.

Sector not found error writing drive (x:)
[MS-DOS]

See "Disk and Device Errors," later in this chapter.

Sector size too large in file (*filename*)
[MS-DOS]

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.

Seek error reading drive (x:)
[MS-DOS]

See "Disk and Device Errors," later in this chapter.

Seek error writing drive (x:)
[MS-DOS]

See "Disk and Device Errors," later in this chapter.

SHARE already installed

[SHARE]

You can install SHARE only once.

Sharing violation reading drive (x:)

[MS-DOS]

See “Disk and Device Errors,” later in this chapter.

SORT: Incorrect DOS version

[SORT]

SORT cannot run on MS-DOS versions prior to MS-DOS 2.0.

SORT: Insufficient disk space

[SORT]

The disk is full.

SORT: Insufficient memory

[SORT]

There is not enough memory to run the SORT program.

Source and target diskettes are not the same format.

Cannot do the copy

[DISKCOPY]

The source and target diskettes must be of the same size and type. For example, you cannot copy from a single-sided diskette to a double-sided diskette. Reformat the target diskette to be of the same type as the source diskette.

Source path required

[REPLACE]

You must specify a source pathname with the REPLACE command.

**Specified drive does not exist,
or is non-removable**

[DISKCOMP]

Specify the name of a valid floppy disk drive. You cannot compare hard disks.

Specified MS-DOS search directory bad
[MS-DOS]

The SHELL command in the CONFIG.SYS file is incorrect. Either COMMAND.COM is not in the place you indicated, or that place does not exist.

Strike a key when ready...
[MS-DOS]

This prompt occurs during command processing. It is always accompanied by another message. It appears if you have inserted a PAUSE command in a batch file. Usually, you are asked to insert disks into appropriate drives before this prompt. Press the space bar to begin command processing.

Symbol defined more than once
[LINK]

The linker found two or more modules that define one symbol name.

Symbol table capacity exceeded
[LINK]

The number or length of the names caused them to exceed the limit of approximately 25K bytes.

Syntax error
[MS-DOS] [FIND]

Be sure you type the command correctly.

System transferred
[FORMAT] [SYS]

The system files have been transferred during FORMAT or SYS command processing.

Terminate batch job (Y/N)?
[MS-DOS]

If you press **[CTRL] [C]** while in batch mode, MS-DOS asks whether or not you wish to end batch file processing. Press **[Y]** (for yes) to end processing. Press **[N]** (for no) to continue.

Too many external symbols in one module
[LINK]

The limit is 256 external symbols per module.

Too many files open

[EDLIN]

MS-DOS cannot open the file to edit on the .bak file because of a lack of system file handles. Increase the number of file handles in the FILES command in the CONFIG.SYS file.

Too many groups

[LINK]

The limit is 10 groups.

Too many libraries specified

[LINK]

The limit is 16 libraries.

Too many public symbols

[LINK]

The limit is 1024 public symbols.

Too many segments or classes

[LINK]

The limit is 256 (segments and classes taken together).

Track 0 bad - disk unusable

[FORMAT]

The FORMAT command can accommodate for defective sectors on the disk, except for those near the beginning. Use another disk.

Unable to create a directory

[MS-DOS] [XCOPY]

MS-DOS cannot create the directory you specified. There might be a name conflict (you have another file of the same name), and or the disk might be full.

Unexpected DOS Error (*nnn*)

[REPLACE]

An unexpected error occurred, where *nnn* is the MS-DOS error number.

Unrecognized command in CONFIG.SYS
[MS-DOS]

There is an invalid command in your CONFIG.SYS file. Refer to Appendix C for valid statements.

Unrecoverable error in directory
Convert directory to file (Y/N)?
[CHKDSK]

If you press ☐ (for yes) in response to this prompt, CHKDSK converts the bad directory into a file. Then, you can either fix or delete the directory.

Unresolved externals: list
[LINK]

The external symbols listed have no defining module among the module or library files specified.

usage: fc [/a] [/b] [/c] [/l] [/lb n] [/w]
[t] [/n] [/NNNN] file1 file2
[FC]

You specified an invalid switch with FC.

VERIFY is off (or on)
[MS-DOS]

This message tells you the current setting of the VERIFY command.

VM read error
[LINK]

This is a disk error; it is not caused by the linker.

Volume in drive (x:) has no label
[MS-DOS] [LABEL]

The screen displays this message whenever you type the DIR or LABEL command for a disk that has no label.

Volume in drive (x:) is (label)
[MS-DOS] [LABEL]

The screen displays this message whenever you type the DIR or LABEL command for a disk that has a label.

Volume label (11 characters, ENTER for none)?
[FORMAT] [LABEL]

The screen displays this message if you enter the LABEL command or if you use the /V switch with FORMAT. Enter a volume label, or press **[ENTER]** to indicate that you don't want to label the disk.

**WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE x WILL BE LOST!**
Proceed with Format (Y/N)?
[FORMAT]

The hard disk you are trying to format contains data. If you want to lose the data and format the disk, press **[Y]** (for yes). If you do not want the files on the disk erased, press **[N]** (for no). Then, copy the files to floppy diskettes, and repeat the FORMAT command.

Warning - directory full
[RECOVER]

The ROOT directory contains too many files for you to use RECOVER. Delete some files in the ROOT directory to free some space.

Warning: no stack segment
[LINK]

None of the object modules specified contains a statement allocating stack space, but you typed the /STACK switch.

Warning: segment of absolute or unknown type
[LINK]

A bad object module exists or an attempt has been made to link modules that the linker cannot handle (for example, an absolute object module).

Warning: Read error in EXE file
[EXE2BIN]

The amount read was less than the size of the header. This is a warning message only.

Write error in tmp file
[LINK]

No disk space remains in which to expand the VM.TMP file.

Write error on run file

[LINK]

This usually indicates there is not enough disk space for the run file.

Write fault error writing drive (x):

[MS-DOS]

See "Disk and Device Errors."

Write protect error writing drive (x):

[MS-DOS]

See "Disk and Device Errors."

Disk and Device Errors

Disk and device errors are errors related to device operation. They come from problems with your computer and its peripherals (such as disk drives, hard disks, diskettes, modems, and printers).

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

```
type action device name
Abort, Retry, Ignore?
```

type refers to one of the following types of errors:

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

action is either READING or WRITING.

device is the equipment that is experiencing the difficulty, such as drive (for disk drive) or PRN (for printer).

name is the letter of the drive (if *device* is a drive) in which the error occurred.

After displaying the error, MS-DOS waits for you to respond to the prompt by pressing one of the following keys:

- ☐ **A** to cancel (abort) the program.
- ☐ **I** to ignore the error and pretend the error did not occur.
- ☐ **R** to repeat the operation. Use this response if you have corrected the error (for example, if you have removed the write-protect tab from a disk to which you are trying to write).

Usually, you will want to recover by first pressing ☐ **R** to try again and—if that fails—by pressing ☐ **A** to cancel the operation. Then, you can take steps to discover the cause of the error.

Another message that might be related to a fault disk read or write operation is:

File allocation table bad for drive x

This message means that the memory copy of one of the allocation tables has pointers to nonexistent blocks. Possibly, the disk or diskette is incorrectly formatted or is not formatted at all. If this error persists, the disk or diskette is unusable as is. You must reformat it before using it again.

GLOSSARY

Following is a list of terms, used in this manual, that might not be familiar to you.

application program	A computer program used to perform a specified task such as word processing or file handling.
argument	A command variable that defines or directs the command in accomplishing its task.
ASCII	American Standard Code for Information Interchange. A universal numeric code for alphanumeric and punctuation characters that allows the exchange of information between computers that are equipped for communication.
assemble	The operation of a program on assembly language source code to create a machine language or <i>binary</i> file. Binary files are given the extension .exe in MS-DOS, and you can execute them by typing the program name and pressing ENTER .
Assembler	A program to create binary files from assembly language source code. (See <i>assemble</i> .)
background	A multitask operation, such as printing files (in the background) while entering data.
batch files	A method of combining several commands into one file that, when called, automatically executes all the commands in series.
binary	Numbers or code to the base of 2. Your computer's central processing unit (CPU) requires binary code for all of its operations; thus, a binary file is often referred to as a machine language program.

buffer	A temporary storage area, usually in the computer's memory or on disk. Your computer has buffers for such tasks as reading and writing to disk and for storing text being sent to a printer.
command	Instructions you give to your computer. Type the command name, any parameters, and press <code>ENTER</code> .
CON	A device name in MS-DOS referring to the console (screen or keyboard).
current directory	The directory in which you are currently operating. See Chapter 2 for information on changing the current directory.
current drive	The operating drive (the drive that contains the disk you are using). See Chapter 2 for information on changing the current drive.
cursor	The screen prompt symbol. In MS-DOS, it is a flashing underscore.
default	An argument or parameter that is <i>built into</i> a command. It is automatically selected if you do not specify another argument or parameter.
delimiter	A symbol that indicates a junction. When used with MS-DOS commands, a delimiter is a space, comma, semicolon, equal sign, vertical bar, or tab.
device	A peripheral piece of equipment attached to your computer, such as a printer, disk drive, or monitor.
directory	A disk area that keeps track of your files. Each disk can have several directories, each of which can contain several subdirectories or files or both.

disk file	A disk area in which you store information.
display	Text or graphics that appears on the video screen of your computer.
drive	The mechanical portion of your computer's magnetic storage device.
dummy argument	A command parameter that is to be replaced before the command can execute successfully. The replacement parameter or value can come from a program or from operator input.
dump	Send data or information to a device. You can dump a screen (text on the monitor) to a disk or dump a buffer to a printer.
error	Text on the screen that indicates a problem. See Section 6 for a description of error messages.
ext or extension	An appendage to a filename, one to three characters long, that provides further identification. Extensions often indicate a type of file such as Maillist.bas, where .bas is the extension and indicates a BASIC file.
external commands	Procedures or utilities not resident in memory. Before MS-DOS can execute external commands, it must load them from disk into memory.
filename	The name given to a file. It can be a maximum of eight characters long.
filespec	A filename plus its extension, such as Memos.bas.
filter	A function that performs a specific operation on data (such as sorting or searching) as it flows between devices (such as disk and a printer), or between files.

format	The process of dividing a disk into tracks and sectors and writing on it the system information needed to organize and store data.
input	Information or data being received by a device.
internal	An MS-DOS utility or procedure that, when the system is loaded, resides in memory. You can execute an internal command even after you remove the system disk from the drive.
LPT	A device name in MS-DOS that refers to the printer.
output	Information or data being sent to a device.
parameter	A variable item of information, appended to a command, that defines or customizes the command.
path	The route to a specific drive and through directories and subdirectories to a specific file.
pathname	The direction the operating system takes to get to a file's location on disk. MS-DOS pathnames have the general form <i>drive:\path\filename.ext</i> .
peripheral	An external device attached to your computer, such as a printer, monitor, keyboard, or disk drive.
piping	A method of making one command's output another command's input, allowing the <i>chaining</i> of commands.
program	See "application program."
PRN	A device name in MS-DOS that refers to the printer.
replaceable parameters	Information appended to a command, usually represented by a symbol such as %, to be replaced by user input when the command is called. See "dummy argument."

ROOT	The first directory level on any disk. All other directories are subdirectories of the ROOT.
string	A specified group of characters, for example, "Hello".
system	Usually refers to your DOS (disk operating system). It can also refer to your computer system, including monitor, CPU, drives, and peripheral equipment.
system prompt	The display on the video screen (A>, B>, C>, or D>) that indicates which drive is the current drive and that tells you MS-DOS is waiting for a command.
switch	A command parameter that activates a special function. The command DIR /W causes the directory listing to be displayed on the screen in the <i>wide</i> mode.
syntax	The rules governing the structure of a command, such as spelling, the order in which commands and parameters are to be given, and which characters are legal delimiters.
template	A storage area that contains the last MS-DOS command used. You can call up the command from the template for editing and re-execution.
toggle	To switch a function on or off. For example, pressing CTRL PRTSC once sends screen input to the printer. Pressing CTRL PRTSC again turns off the function.
track	A concentric circle on a disk on which data is stored. Each track is divided into several sectors.

verify	A function that checks data to be sure that it is copied or stored correctly.
wild card	One or more symbols that are used to represent one or more variable characters. For example, you can use the wild card ? in a filename to <i>stand in place of</i> a legal filename character.

ASCII AND SCAN CODES

The following table lists the keys, in scan code order, and the ASCII codes generated by each (which depends on the shift status). The entries in the table are:

- **SCAN CODE** — a value in the range 01H-54H (hexadecimal) that uniquely describes which key is pressed.
- **NORMAL CASE** — the normal (unshifted) ASCII value (returned when only the indicated key is pressed).
- **UPPER CASE** — the shifted ASCII value (returned when **SHIFT** is also pressed).
- **CTRL CASE** — the control ASCII value (returned when **CTRL** is also pressed).
- **ALT CASE** — the alternate ASCII value (returned when **ALT** is also pressed).

All numeric values in the table are expressed in hexadecimal. Those values preceded by an X are extended ASCII codes (they are preceded by an ASCII NUL [= 00]).

A marking of — indicates that no ASCII code is generated. A marking of ** indicates that no ASCII code is generated and that, instead, the special function is performed.

Appendix B / ASCII and Scan Codes

Scan code to ASCII translations for US English Tandy 3000 keyboard

key # - SCAN CODE	NORM CASE (ASCII code)		UPPER CASE (ASCII code)		CTRL CASE (ASCII code)		ALT CASE (ASCII code)	
01	ESC	1B	ESC	1B	ESC	1B	—	—
02	1	31	!	21	—	—	ALT1	X078
03	2	32	@	40	NULL	00	ALT2	X079
04	3	33	#	23	—	—	ALT3	X07A
05	4	34	\$	24	—	—	ALT4	X07B
06	5	35	%	25	—	—	ALT5	X07C
07	6	36	^	5E	RS	1E	ALT6	X07D
08	7	37	&	26	—	—	ALT7	X07E
09	8	38	*	2A	—	—	ALT8	X07F
0A	9	39	(28	—	—	ALT9	X080
0B	0	30)	29	—	—	ALT0	X081
0C	-	2D	—	5F	US	1F	ALT-	X082
0D	=	3D	+	2B	—	—	ALT=	X083
0E	BS	08	BS	08	DEL	7F	—	—
0F	→	09	←	X00F	—	—	—	—
10	q	71	Q	51	DC1	11	ALTQ	X010
11	w	77	W	57	ETB	17	ALTW	X011
12	e	65	E	45	ENQ	05	ALTE	X012
13	r	72	R	52	DC2	12	ALTR	X013
14	t	74	T	54	DC4	14	ALTT	X014
15	y	79	Y	59	EM	19	ALTY	X015
16	u	75	U	55	NAK	15	ALTU	X016
17	i	69	I	49	HT	09	ALTI	X017
18	o	6F	O	4F	SI	0F	ALTO	X018
19	p	70	P	50	DLE	10	ALTP	X019
1A	[5B	{	7B	ESC	1B	—	—
1B]	5D	}	7D	GS	1D	—	—
1C	CR	0D	CR	0D	LF	0A	—	—
1D	CTRL	—	CTRL	—	CTRL	—	CTRL	—
1E	a	61	A	41	SOH	01	ALTA	X01E
1F	s	73	S	53	DC3	13	ALTs	X01F
20	d	64	D	44	EOT	04	ALTD	X020
21	f	66	F	46	ACK	06	ALTF	X021
22	g	67	G	47	BEL	07	ALTG	X022
23	h	68	H	48	BS	08	ALTH	X023
24	j	6A	J	4A	LF	0A	ALTJ	X024
25	k	6B	K	4B	VT	0B	ALTK	X025
26	l	6C	L	4C	FF	0C	ALTL	X026
27	;	3B	:	3A	—	—	—	—
28	'	27	"	22	—	—	—	—
29	`	60	~	7E	—	—	—	—
2A	left SHIFT	—	left SHIFT	—	left SHIFT	—	left SHIFT	—
2B	\	5C		7C	FS	1C	—	—
2C	z	7A	Z	5A	SUB	1A	ALTZ	X02C
2D	x	78	X	58	CAN	18	ALTx	X02D
2E	c	63	C	43	ETX	03	ALTC	X02E
2F	v	76	V	56	SYN	16	ALTv	X02F
30	b	62	B	42	STX	02	ALTB	X030
31	n	6E	N	4E	SO	0E	ALTN	X031
32	m	6D	M	4D	CR	0D	ALTM	X032

key # -	SCAN CODE	NORM CASE (ASCII code)	UPPER CASE (ASCII code)	CTRL CASE (ASCII code)	ALT CASE (ASCII code)			
33	,	2C	<	3C	— —			
34	.	2E	>	3E	— —			
35	/	2F	?	3F	— —			
36	right SHIFT	—	right SHIFT	—	right SHIFT			
37	*	2A	PrScr**	CPrScr** X072	— —			
38	ALT	—	ALT	—	ALT			
39	SPACE	20	SPACE	20	SPACE X020			
3A	CAPS	—	CAPS	—	CAPS			
3B	F1	X03B	F11	X054	F31	X068		
3C	F2	X03C	F12	X055	F22	X05F	F32	X069
3D	F3	X03D	F13	X056	F23	X060	F33	X06A
3E	F4	X03E	F14	X057	F24	X061	F34	X06B
3F	F5	X03F	F15	X058	F25	X062	F35	X06C
40	F6	X040	F16	X059	F26	X063	F36	X06D
41	F7	X041	F17	X05A	F27	X064	F37	X06E
42	F8	X042	F18	X05B	F28	X065	F38	X06F
43	F9	X043	F19	X05C	F29	X066	F39	X070
44	F10	X044	F20	X05D	F30	X067	F40	X071
45	NUM LOCK	—	NUM LOCK	—	PAUSE **	NUM LOCK	—	
46	SCROLL LOCK	—	SCROLL LOCK	—	BREAK **	SCROLL LOCK	—	
47	7	37	HOME	X047	CLR SCN	X077	†	—
48	8	38	↑	X048	—	—	†	—
49	9	39	PG UP	X049	TOP OF TEXT AND HOME	X084	†	—
4A	—	2D	—	2D	—	—	—	—
4B	4	34	←	X04B	LEFT	X073	†	—
4C	5	35	—	—	ONE WORD	—	†	—
4D	6	36	→	X04D	RIGHT	X074	†	—
4E	+	2B	+	2B	—	—	—	—
4F	1	31	END	X04F	ERASE	X075	†	—
50	2	32	↓	X050	TO EOL	—	†	—
51	3	33	PG DN	X051	ERASE	X076	†	—
52	0	30	INS	X052	TO EOS	—	†	—
53	.	2E	DEL	X053	—	—	—	—
54	SYS**	—	SYS**	—	SYS**	—	SYS**	—

Note: When the NUMLOCK light is on and [SHIFT] is pressed, the BASE CASE characters are produced.

— indicates that no ASCII code is generated for the key combination.

X values preceded by X are extended ASCII codes. The ROM BIOS returns a 0 ASCII code and the number in the table for the scan code.

† The [ALT] key provides a way to generate the ASCII code of decimal numbers in the range 1 to 255. Hold down the [ALT] key while typing a number in that range. When you release [ALT], the character of the ASCII code you typed is generated and displayed.

ASCII and Scan Code Special Handling

CTRL BREAK break	empties the keyboard queue and executes the keyboard break interrupt (int 1BH). Places a NULL ASCII scan code in the keyboard queue.
CTRL S pause	delays system activity (except external interrupts) until you press another key.
SHIFT PRSCR PrScr	Invokes the BIOS print screen function (int 5H). A second PrScr halts the printer output.
CTRL PRSCR CPrScr	Tells MS-DOS to direct console output to both the printer and the console. A second CPrScr halts printer output.

ASCII Character Codes

The ASCII and Scan Codes table listed the ASCII codes (in hexadecimal) generated by each key. This table lists the characters generated by those ASCII codes.

Note: All ASCII codes in this table are expressed in decimal form.

You can display the characters listed by doing either of the following:

- Using the BASIC statement `PRINT CHR$(code)`, where *code* is the ASCII code.
- Pressing **ALT** and, without releasing it, typing the ASCII code on the numeric keypad.

For Codes 0-31, the table also lists the standard interpretations. The interpretations are usually used for control functions or communications.

Note: The BASIC program editor has its own special interpretation of some codes and might not display the character listed.

ASCII CHARACTER CODES

Chr	Dec	Hex	Chr	Dec	Hex
NUL	000	00H	SPACE	032	20H
OH	001	01H	!	033	21H
STX	002	02H	"	034	22H
ETX	003	03H	#	035	23H
EOT	004	04H	\$	036	24H
ENQ	005	05H	%	037	25H
ACK	006	06H	&	038	26H
BEL	007	07H	'	039	27H
BS	008	08H	(040	28H
HT	009	09H)	041	29H
LF	010	0AH	*	042	2AH
VT	011	0BH	+	043	2BH
FF	012	0CH	,	044	2CH
CR	013	0DH	-	045	2DH
SO	014	0EH	.	046	2EH
SI	015	0FH	/	047	2FH
DLE	016	10H	0	048	30H
DC1	017	11H	1	049	31H
DC2	018	12H	2	050	32H
DC3	019	13H	3	051	33H
DC4	020	14H	4	052	34H
NAK	021	15H	5	053	35H
SYN	022	16H	6	054	36H
ETB	023	17H	7	055	37H
CAN	024	18H	8	056	38H
EM	025	19H	6	057	39H
SUB	026	1AH	:	058	3AH
ESCAPE	027	1BH	;	059	3BH
FS	028	1CH	<	060	3CH
GS	029	1DH	=	061	3DH
RS	030	1EH	>	062	3EH
US	031	1FH	?	063	3FH

Appendix B / ASCII and Scan Codes

Chr	Dec	Hex	Chr	Dec	Hex
@	064	40H	'	096	60H
A	065	41H	a	097	61H
B	066	42H	b	098	62H
C	067	43H	c	099	63H
D	068	44H	d	100	64H
E	069	45H	e	101	65H
F	070	46H	f	102	66H
G	071	47H	g	103	67H
H	072	48H	h	104	68H
I	073	49H	i	105	69H
J	074	4AH	j	106	6AH
K	075	4BH	k	107	6BH
L	076	4CH	l	108	6CH
M	077	4DH	m	109	6DH
N	078	4EH	n	110	6EH
O	079	4FH	o	111	6FH
P	080	50H	p	112	70H
Q	081	51H	q	113	71H
R	082	52H	r	114	72H
S	083	53H	s	115	73H
T	084	54H	t	116	74H
U	085	55H	u	117	75H
V	086	56H	v	118	76H
W	087	57H	w	119	77H
X	088	58H	x	120	78H
Y	089	59H	y	121	79H
Z	090	5AH	z	122	7AH
[091	5BH	{	123	7BH
\	092	5CH		124	7CH
]	093	5DH	}	125	7DH
^	094	5EH	-	126	7EH
_	095	5FH	DEL	127	7FH

Dec = decimal, Hexadecimal(H), CHR = character,
LF = Line Feed, FF = Form Feed, CR = Carriage Return,
DEL = Rub out

ASCII CHARACTER CODES

ASCII Code	Character	Control Character
000	(null)	NUL
001	☺	SOH
002	☹	STX
003	♥	ETX
004	♦	EOT
005	♣	ENQ
006	●	ACK
007	(beep)	BEL
008	■	BS
009	(tab)	HT
010	(line feed)	LF
011	(home)	VT
012	(form feed)	FF
013	(carriage return)	CR
014	🎵	SO
015	⚙	SI
016	▶	DLE
017	◀	DC1
018	↕	DC2
019	!!	DC3
020	¶	DC4
021	§	NAK
022	—	SYN
023	⏏	ETB
024	↑	CAN
025	↓	EM
026	→	SUB
027	←	ESC
028	(cursor right)	FS
029	(cursor left)	GS
030	(cursor up)	RS
031	(cursor down)	US

ASCII CHARACTER CODES

ASCII Code	Character	ASCII Code	Character
032	(space)	068	D
033	!	069	E
034	''	070	F
035	#	071	G
036	\$	072	H
037	%	073	I
038	&	074	J
039	,	075	K
040	(076	L
041)	077	M
042	*	078	N
043	+	079	O
044	'	080	P
045	-	081	Q
046	.	082	R
047	/	083	S
048	0	084	T
049	1	085	U
050	2	086	V
051	3	087	W
052	4	088	X
053	5	089	Y
054	6	090	Z
055	7	091	[
056	8	092	\
057	9	093]
058	:	094	^
059	;	095	_
060	<	096	:
061	=	097	a
062	>	098	b
063	?	099	c
064	@	100	d
065	A	101	e
066	B	102	f
067	C	103	g

ASCII CHARACTER CODES

ASCII Code	Character	ASCII Code	Character
104	h	140	î
105	i	141	ï
106	j	142	Ä
107	k	143	Å
108	l	144	Ê
109	m	145	ë
110	n	146	Ë
111	o	147	ô
112	p	148	Ö
113	q	149	ò
114	r	150	û
115	s	151	ù
116	t	152	ÿ
117	u	153	Ö
118	v	154	Ü
119	w	155	€
120	x	156	£
121	y	157	¥
122	z	158	Pl
123	{	159	f
124		160	á
125	}	161	í
126	~	162	ó
127	☐	163	û
128	Ç	164	ñ
129	ü	165	Ñ
130	é	166	ä
131	â	167	o
132	ä	168	ç
133	·ä	169	┐
134	°ä	170	└
135	ç	171	½
136	ê	172	¼
137	ë	173	ì
138	è	174	«
139	ï	175	»

ASCII CHARACTER CODES

ASCII Code	Character	ASCII Code	Character
176	⌞	212	⌞
177	⌟	213	⌟
178	⌠	214	⌠
179	⌡	215	⌡
180	⌢	216	⌢
181	⌣	217	⌣
182	⌤	218	⌤
183	⌥	219	■
184	⌦	220	■
185	⌧	221	■
186	⌨	222	■
187	〈	223	■
188	〉	224	α
189	⌫	225	β
190	⌬	226	Γ
191	⌭	227	π
192	⌮	228	Σ
193	⌯	229	σ
194	⌰	230	μ
195	⌱	231	τ
196	⌲	232	Φ
197	⌳	233	Θ
198	⌴	234	Ω
199	⌵	235	δ
200	⌶	236	∞
201	⌷	237	Ø
202	⌸	238	€
203	⌹	239	∩
204	⌺	240	≡
205	⌻	241	±
206	⌼	242	≥
207	⌽	243	≤
208	⌾	244	∫
209	⌿	245	∫
210	⌿	246	÷
211	⌿	247	≈

ASCII CHARACTER CODES

ASCII Code	Character	ASCII Code	Character
248	°	252	η
249	●	253	²
250	•	254	■
251	√	255	(blank 'FF')

CONFIGURING YOUR SYSTEM

To configure your system to accommodate various peripheral devices, you can create a file named CONFIG.SYS. This is an ASCII file of commands that you want MS-DOS to execute on startup. If CONFIG.SYS is present in the ROOT directory of your system disk, MS-DOS automatically reads the file during boot and executes the commands it finds there.

If there is no CONFIG.SYS file on your MS-DOS system disk, you can use EDLIN to create the file. (See Part 3 for information on creating files with EDLIN.) You can also create a CONFIG.SYS file by using the COPY command in this format:

```
COPY con config.sys 
```

The cursor blinks as the system waits for you to enter text—in this case, the CONFIG.SYS commands. Regardless of how you create the file, press to end input and to save on disk the lines you typed. Again, be sure to save CONFIG.SYS in your system ROOT directory so that MS-DOS can execute it on startup.

Because MS-DOS executes CONFIG.SYS only at startup, MS-DOS does not recognize new commands in CONFIG.SYS until you reboot. Therefore, always reboot the computer each time you add commands to the file.

CONFIG.SYS Commands

Here is a summary of the commands you can use in your CONFIG.SYS file. Following the summary is a detailed explanation of each command. The appendix concludes with a sample CONFIG.SYS file. (See Appendix D for information about the various device drivers you can install, using the DEVICE command.)

Command	Purpose
---------	---------

BREAK	Sets the <input type="checkbox"/> CTRL <input type="checkbox"/> C check.
BUFFERS	Sets the number of sector buffers.
COUNTRY	Allows for international time, date, and currency format, and characters.
DEVICE	Installs a <i>device driver</i> into the system. Include one DEVICE command for each driver you install.
DRIVPARM	Defines parameters for block devices.
FCBS	Specifies the number of FCBs (<i>file control blocks</i>) that you can open at one time.
FILES	Sets the number of open files that can access certain MS-DOS system calls.
LASTDRIVE	Sets the maximum number of drives you can access.
SHELL	Causes MS-DOS to start a command processor other than the standard COMMAND.COM processor.

BREAK

BREAK [ON | OFF]

Sets the **CTRL C** check.

Depending on the program you are running, you might use **CTRL C** to stop an activity such as file sort. Normally, MS-DOS checks to see if you typed **CTRL C** while it was writing to the screen or printer, reading from the keyboard, or performing asynchronous communications. When that is the case, you cannot stop execution of a program unless it is performing one of those functions. Setting **BREAK** to **ON** lets you extend the **CTRL C** check to other functions, such as disk reads and writes. Setting **BREAK** to **OFF** re-establishes the system default (**OFF**).

Example: **BREAK=ON**

extends the **CTRL C** check.

BUFFERS

BUFFERS = *number*

Lets you set the number of *disk buffers* that MS-DOS allocates in memory at the time you start the system.

A disk buffer is a block of memory in which MS-DOS temporarily stores data during disk input/output operations whenever the amount of data read or written is not an exact multiple of the sector size. Buffers are 512 bytes long, and the default number of buffers is two.

Increasing the number of buffers can increase the speed of operations, as data still resident in the buffers can be read without additional disk drive access. (The greater the number of buffers, the greater the block of data available in memory.) If you create many subdirectories, you can increase efficiency by using a number in the range 20 to 30. For application programs such as word processors, you might find that a number in the range 10 to 20 provides the best performance. Experiment to find the ideal number to use, or consult your application program manual for the requirements of a specific program. The maximum number of buffers allowed is 99.

Example: **BUFFERS=10**

sets the number of disk buffers to 10.

COUNTRY

COUNTRY = *country code*

Enables MS-DOS to use the date and time format and to produce the currency symbols for any of several countries.

The following tables lists the valid country codes. The default is 001 (for the United States).

Country	Country Code
Australia	061
Belgium	032
Canadian-French	002
Denmark	045
Finland	358
France	033
Germany	049
Italy	039
Israel	972
Middle East	785
Netherlands	031
Norway	047
Portugal	351
Spain	034
Sweden	046
Switzerland	041
United Kingdom	044
United States	001

Example: COUNTRY=033

sets the country code to 033 (France).

DEVICE

DEVICE = *pathname*

Installs the device driver specified by *pathname* into the system list of drivers.

Your system is designed to provide a number of standard screen, keyboard, printer, and disk drive functions without special instructions. By specifying device drivers, you can also enable it to perform a wide range of non-standard functions with various peripheral devices (including printers, modems, and monitors). For example, you can enable it to use external drives by installing the DRIVER.SYS device driver.

DRIVER.SYS is only one of several device drivers already written for you and provided on your MS-DOS system diskette or Supplemental Programs diskette.

The DEVICE command tells MS-DOS to load a particular device driver automatically whenever you start the system. For each driver you want to install, you need to include one DEVICE command in your CONFIG.SYS file. See Appendix D for the command needed for each of the drivers provided. If you have written your own device drivers, include DEVICE commands for them, also.

Example: DEVICE=B:HDRIVE.SYS

installs HDRIVE.SYS (the driver that lets you use non-standard hard disks with your system) from the ROOT directory of the diskette in Drive B.

Note: Because MS-DOS installs the driver each time you start up the system, it is simplest to have the driver on the current system disk. Then, you don't have to specify (in the DEVICE command) where to look for the driver.

DRIVPARM

DRIVPARM = /D:*drive* [/C] [/F:*form factor*]
[/H:*head*] [/N] [/S:*sectors*] [/T:*tracks*]

Lets you define parameters for block devices when you start MS-DOS, overriding the original MS-DOS device driver settings.

- /D specifies the logical drive number. *drive* is a value in the range 0 to 255. Drive A=0, Drive B=1, Drive C=2, and so on.
- /C specifies that *changeline* (doorlock) support is required.
- /F specifies the form factor index, where:
 - 0 = 320/360K floppy
 - 1 = 1.2M floppy
 - 2 = 720K floppy (3½-inch drive)
 - 3 = 8-inch single-density floppy
 - 4 = 8-inch double-density floppy
 - 5 = hard disk
 - 6 = tape drive
 - 7 = other

If you omit /F, DRIVPARM uses a default of 2 (720K).

- /H specifies the maximum head number. *head* is a value in the range 1 to 99.
- /N tells MS-DOS that the device is non-removable.
- /S specifies the number of sectors per track. *sectors* is a value in the range 1 to 99.
- /T specifies the number of tracks per side. *tracks* is a value in the range 1 to 999.

Example: You might have a computer with an internal tape drive unit on Drive D that is configured at boot time to write 20 tracks of 40 sectors per track. If you want to reconfigure this tape drive to write 10 tracks of 99 sectors each, you can put the following line in your CONFIG.SYS file:

```
DRIVPARM=/D:3 /F:6 /H:1 /S:99 /T:10
```

This overrides the default device driver settings, and supports a tape drive as Drive D. (In this case, the logical and physical drive numbers are identical.) This tape drive has one head, and supports a format of 10 tracks and 99 sectors per track (assuming that the device driver for the tape drive also supports this configuration). You might want to use this method to create a tape that you can read on another computer that reads only this format.

```
DRIVPARM=/D:1
```

Drive B is a 3½-inch floppy disk drive. Note that you can omit the /F parameter because it defaults to 2 (a 3½-inch drive).

FCBS

FCBS *n1,n2*

Sets the number of files opened by file control blocks that can be opened at any one time, and lets you specify a number of these to protect from inadvertent closure.

n1 specifies the number of files opened by FCBs that can be opened at any one time. *n1* can be in the range 0 to 255. The default value is 4.

n2 specifies the number of files opened by FCBs that MS-DOS cannot close automatically if an application tries to open more than *n1* files by FCB. This option protects the first *n2* files opened by FCBs from being closed. *n2* can be in the range 0 to 255. The default value is 0.

Example: FCBS=4,2

specifies that four files can be open, and protects two files.

FILES

FILES = *number*

Sets the number of open file handles compatible with XENIX® that the MS-DOS system calls can access.

number is the number of open file handles that the system calls can access. System calls in the range 2FH to 60H are compatible with the XENIX operating system.

number can be any value in the range 8 to 255. The default value is 8.

Example: **FILES=20**

sets the number of accessible open handles to 20.

LASTDRIVE

LASTDRIVE = *drive*

Sets the last valid drive designation that MS-DOS accepts. This command is only useful in a network environment.

drive is the logical drive specification of the last valid drive. It is a letter in the range A to Z. Note that you do not place a colon following the drive letter. The default value is E.

At startup, MS-DOS recognizes five drive letters, A-E, regardless of the number of physical drives you have on your system. A network redirection must occur to make any of the extra drives defined by LASTDRIVE valid.

MS-DOS allocates a data structure for each drive you specify. So, do not specify more than necessary.

Example: LASTDRIVE=M

sets the last drive to Drive "M", unless you have added an external logical device with the device driver DRIVER.SYS. MS-DOS now recognizes Drives A through M. (See Appendix D, "Installable Device Drivers," for more information on DRIVER.SYS.)

SHELL

SHELL = *pathname*

Causes MS-DOS to start a command processor other than the standard COMMAND.COM processor.

This command is intended for system programmers who write their own command processor.

Example: **SHELL**=\BIN\COMMAND.COM /E:3000 A:\BIN /P

uses the COMMAND.COM shell in the \BIN directory with an environment size of 3000 bytes.

Sample CONFIG.SYS File

A typical CONFIG.SYS file might look like this:

```
BUFFERS=10
FILES=10
DEVICE=\BIN\NETWORK.SYS
BREAK=ON
SHELL=A:\BIN\COMMAND.COM /E:3000 A:\BIN /P
LASTDRIVE=E
```

The above file sets the number of disk buffers to 10 and the number of open file handles to 10. It also tells MS-DOS to search for the pathname \BIN\NETWORK to find the device you are adding to the system. The file is usually supplied on disk with your device. Be sure you save the device file in the directory that you specify with the DEVICE command.

The file also sets the MS-DOS command EXEC to the COMMAND.COM file located on disk in Drive A in the \BIN directory. The /E switch sets the size of the environment to 3000 bytes. The A:\BIN tells COMMAND.COM where to look for itself when it needs to be reread from disk. The /P switch tells COMMAND.COM that it is the first program running on the system so that it can process the MS-DOS EXIT command. (See COMMAND in Part 2 for more information on the command processor.)

The last logical drive on the system is Drive E, unless you have added an external logical device, using DRIVER.SYS. (See Appendix D for more information on DRIVER.SYS.)

INSTALLABLE DEVICE DRIVERS

Your MS-DOS system diskette contains four device drivers that give your computer special abilities. These drivers are: LPDRVER.SYS, VDISK.SYS, ANSL.SYS, and DRIVER.SYS. Your Supplemental Programs diskette contains three other drivers: HDRIVE.SYS, SPOOLER.SYS, and MLPART.SYS. Following is a list of the drivers and their functions.

Device Driver	Purpose
ANSL.SYS	Provides programmers with many extended screen and keyboard features, such as the ability to change graphics functions.
DRIVER.SYS	Enables your system to support more than two floppy disk drives and more than two hard disk drives. Also, DRIVER.SYS lets you assign more than one logical drive letter to a single drive.
HDRIVE.SYS	Lets you use an extended range of hard disks with your computer.
LPDRVR.SYS	Lets you configure your system to take full advantage of your printer's abilities.
MLPART.SYS	Enables you to access multiple MS-DOS partitions on a hard disk drive. You need to use MLPART.SYS if you want to take advantage of more than the first 32 megabytes of storage on one hard disk.
MODEVM.SYS	(For use only in countries outside the U.S.) Improves video display output on a VM-1 monitor (Cat. No. 26-5111) used with Tandy's Deluxe Text Display Adapter (Cat. No. 25-3046) or Deluxe Graphics Display Adapter (Cat. No. 25-3047).
SPOOLER.SYS	Lets you continue processing data with your computer while using your printer.
VDISK.SYS	Lets you establish a simulated (<i>virtual</i>) disk drive in RAM.

This appendix contains the specifics about all the drivers listed in the preceding chart. The information is presented in the following order:

1. A general description of the device driver.
2. The `DEVICE` command that you need to place in your `CONFIG.SYS` file to install the driver—including any parameters the command requires. The `DEVICE` command tells MS-DOS to load the driver automatically whenever you start the system. Therefore, you use it only once. (See Appendix C, “Configuring Your System,” for information about `CONFIG.SYS`.)
3. Detailed information that you need.

Be sure to read all the information provided on a particular device driver before installing the driver.

Note: The optimum location for your device drivers is your current system disk. There, they are always available for installation by MS-DOS on system startup.

ANSI.SYS

(For Extended Screen and Keyboard Control)

ANSI.SYS is a loadable driver that provides programmers with many extended screen and keyboard features. With it installed, you can change graphics functions, move the cursor, and reassign the meaning of any key on the keyboard by issuing special character sequences from within your program. These sequences are valid only when issued through MS-DOS function calls 1, 2, 6, and 9. For more information, see your *MS-DOS 3.1 Programmer's Reference Manual*.

The DEVICE command that you need to place in CONFIG.SYS to install the ANSI.SYS driver is:

```
DEVICE=ANSI.SYS
```

DRIVER.SYS

(For Additional Disk Drives)

DRIVER.SYS is an installable device driver that enables your system to support more than two floppy disk drives and more than two hard disk drives.

In addition, DRIVER.SYS lets you assign more than one logical drive letter to a single drive. An example of how this can be helpful follows: Suppose your computer has a 5¼-inch floppy disk drive, a 3½-inch floppy disk drive, and a hard disk drive. If you want to copy file Sales.dat from one 3½-inch diskette to another 3½-inch diskette, you need to do a single-drive copy in Drive B, using the command:

```
COPY B:sales.dat B:sales.dat
```

MS-DOS does not allow this, however, because it has no way of knowing that you want to swap the source and target diskettes in Drive B. Instead, it displays a message that reminds you that you cannot copy a file onto itself.

To get around this problem, you can include the following statement in CONFIG.SYS:

```
DEVICE=DRIVER.SYS /D:1 /F:2
```

This command causes MS-DOS to assign the next available drive letter (D) to Drive B (in addition to the letter B), enabling you to then use the command:

```
COPY B:sales.dat D:sales.dat
```

Now, COPY prompts you to swap diskettes as necessary.

The command that you need to place in CONFIG.SYS to load the DRIVER.SYS device driver is:

DEVICE = DRIVER.SYS /D:*drive* /C: /F:*form factor*
 /H:*maximum head* /N /S:*sectors* /T:*tracks*

where:

/D specifies the physical drive number in the range 0 to 255. Floppy drives are numbered starting at 0, and hard disk drives are numbered starting at hexadecimal 80. For example, if you have a computer with two floppy disk drives, the drives are numbered 0 and 1. If you add another floppy disk drive, its physical drive number is hexadecimal 02.

If you have one floppy disk drive and one hard disk drive, the floppy disk drive is hexadecimal 00, but the hard disk drive is hexadecimal 80.

/C specifies that *changeline* (doorlock) support is required. This is a hardware feature that allows the software to know if the drive door is opened.

/F specifies the form factor index, where:

- 0 = 320/360K floppy
- 1 = 1.2M floppy
- 2 = 720K floppy (3½-inch drive)
- 3 = 8-inch single-density floppy
- 4 = 8-inch double-density floppy
- 5 = hard disk
- 6 = tape drive
- 7 = other

If you omit /F, DRIVER.SYS uses a default of 2 (720K).

/H specifies the maximum head number. *head* is a value in the range 1 to 99.

/N tells MS-DOS that the device is non-removable.

/S specifies the number of sectors per track. *sectors* is a value in the range 1 to 99.

/T specifies the number of tracks per side. *tracks* is a value in the range 1 to 999.

HDRIVE.SYS

(For Non-Standard Hard Disks)

HDRIVE.SYS lets you use an extended range of hard disk drive types with your system. HDRIVE's function is to read the drive parameters from your hard disk, and update the system parameters if they do not match.

Immediately after you add any hard disk drive to your system, your first step is run the SETUP program to tell the system the drive type. The SETUP program is on your Utilities diskette.

To determine your drive type, compare the number of cylinders and the number of heads for your drive with the numbers in the following table. (Each hard disk drive comes with the information you need.) If your information exactly matches a type in the table, you have a *standard* drive. If it does not, you have a *non-standard* drive.

Type	Cylinders	Heads
1	306	4
2	615	4
3	615	6
4	940	8
5	940	6
6	615	4
7	462	8
8	733	5
9	900	15
10	820	3
11	855	5
12	855	7
13	306	8
14	733	7
15 Reserved	

If you have a standard hard disk, you can simply specify the type while running SETUP, and you do not need to use HDRIVE. This is because the standard drive types are recorded in permanent CMOS memory in your computer. When finished running SETUP, you can use either the FORMAT HARD DISK utility (also on your Utilities diskette) or the HSECT command (on your MS-DOS system diskette) to format your hard disk's sectors. Then, you can use the FORMAT command to finish preparing your hard disk.

If you have a non-standard hard disk, specify a type that has the same number of heads and fewer cylinders than your drive has. Then, after you run SETUP, use the FORMAT HARD DISK utility (on your Utilities diskette) or the HSECT command (on your MS-DOS system diskette) to format your hard disk's sectors.

After you use FORMAT HARD DISK or HSECT, install the HDRIVE device driver by placing the following command in CONFIG.SYS:

```
DEVICE=HDRIVE.SYS
```

Reboot the computer. Then, use the FDISK and FORMAT commands to finish preparing your hard disk for use.

LPDRVR.SYS

(For Extended Printer Capabilities)

The loadable printer driver provides any Tandy printer with several capabilities. The functions of LPDRVR are:

- Set the number of lines per page
- Set vertical tabs
- Set the form feed function
- Set horizontal tabs
- Tab horizontally
- Set the skip perforation function
- Cancel the skip perforation function
- Ignore the next *n* codes
- Reset the printer driver
- Convert a single code into a series of printer codes
- Repeat *n* characters
- Suppress the line feed function

The **DEVICE** command that you need to place in **CONFIG.SYS** to install the printer driver is:

```
DEVICE=LPDRVR.SYS
```

To use any of the printer functions, you must do the following:

1. Install the printer driver by adding this statement to your **CONFIG.SYS** file:

```
DEVICE=LPDRVR.SYS
```

Do this only once.

2. Set up your printer. If your printer has switches to control carriage returns, line feeds, or both, then set **CR=NL** and **LF=LF** or **NL**. You can also use the **MODE** command to configure LPDRVR to your switch settings.

If printed text is doubled spaced or if one line is overprinting another use **MODE LFOFF** or **MODE LFON** to turn line feeds after a carriage return off or on. See **MODE** for more information.

3. Look up the function's control code sequence in the Printer Control Codes table in this section. Then find the equivalent ASCII code(s) in the ASCII Character Code table in Appendix B. For example, the control code needed to set lines per page is ESCAPE C;*n*. The ASCII equivalent of ESCAPE is 27. The ASCII equivalent of C is 67. *n* is the number of lines.
4. Send the control code sequence, in ASCII form, to the printer driver. You can do this in any of three ways.
 - By using BASIC's LPRINT statement with the CHR\$ function, as described in the BASIC reference manual.
 - By making an MS-DOS function call, as described in the MS-DOS programmer's reference manual.
 - By making a BIOS call, as described in the MS-DOS programmer's reference manual.

The following is an example of using BASIC to set the skip perforation to 5.

```
LPRINT CHR$(27); CHR$(78); CHR$(05)ENTER
```

CHR\$(27) sends the ESCAPE, CHR\$(78) sends the N, and CHR\$(05) sends the number of lines to skip.

PRINTER CONTROL CODES

Function	Code	Result
Set lines per page	ESCAPE C; <i>n</i> ;	Sets the page length to <i>n</i> lines. <i>n</i> is a number in the range of 1 to 127. Lines per page is initially set at 66. Issue this command before setting vertical tabs or form feed.
Set horizontal tabs	ESCAPE D; <i>n1</i> ; <i>n2</i> ; <i>n3</i> ;... <i>nk</i> ;NUL;	Sets horizontal tab stops at <i>n1</i> , <i>n2</i> , <i>n3</i> and so on. The numbers can be in the range 1-131. Tab stops initially are set to every 8 columns. Use ESCAPE D to change them. ESCAPE D;0 resets tabs to the initial state.
Set vertical tabs	ESCAPE B; <i>n1</i> ; <i>n2</i> ; <i>n3</i> ;... <i>NK</i> ;nul;	Sets vertical tab stops to <i>n1</i> , <i>n2</i> , <i>n3</i> , and so on. The numbers can be in the range 1 to the page length minus 1. When the printer is turned on, no tab stops are set, and the printer advances according to line feeds. Use ESCAPE B to set the tabs. ESCAPE B;0 resets tabs to the initial state.
Horizontal Tab	HT	Tabs to the next horizontal tab stop.
Vertical Tab		Tabs down to the next vertical tab stop.

Function	Code	Result
Advance to top of page (form feed)	FF	Advances paper to the next top of page. The printer position is initially top of form. A form feed advances the printer to the top of the next page. To change the number of lines per page use ESCAPE C.
Set skip perforation	ESCAPE N; <i>n</i> ;	Sets the number of lines to skip while printing to <i>n</i> . Use ESCAPE N to put blank lines on the page perforation. Skip is initially set to 0 lines.
Cancel skip perforation	ESCAPE O	Cancels ESCAPE N.
Pass <i>n</i> codes directly to the printer	ESCAPE V; <i>n</i> ;	
Reset (cancel) driver	CAN or DEL	Resets the printer port.
Suppress line feed after carriage return	ESCAPE Y; <i>n</i>	If <i>n</i> is 0, the line feed suppression is turned off. If <i>n</i> is any number greater than 0, the line feed suppression is turned on. Initially line feeds are suppressed.
Repeat char <i>n</i> times	FS; <i>n</i> ; <i>char</i>	Prints a character or string translation <i>n</i> times.
Translate <i>char</i> to string	ESCAPE W; <i>n</i> ; <i>char</i> ; <i>string</i>	Defines a character to string conversion.
Disable LPDRVR	ESCAPE!	Disables all of LPDRVR's capabilities. (See "Notes and Suggestions," below.)

Notes and Suggestions

- Use BIOS function AH = 1, DX = printer (0-2), int 17H to effectively reset LPDRVR and your printer. This resets the LPDRVR mode and allows it to begin a new command or ESC sequence. It also resets the top of form counter. Some programs do this before printing, such as PRINT.COM. This method does not change the form length, tabs, or skip perforation settings.
- When using the BASIC LPRINT command to define character to string translations, the following features can alter the output to LPDRVR and the printer.
 - If the string is longer than WIDTH, BASIC inserts a carriage return and a line feed.
 - BASIC automatically sends a line feed (CHR\$(10)) with a carriage return code (CHR\$(13)).
- You might need to disable LPDRVR when running a word processing program which is correctly configured for your printer. ESCAPE CHR\$(33) disables LPDRVR when it is enabled and enables LPDRVR when it is disabled. Resetting LPDRVR with interrupt 17H as previously described also enables LPDRVR and resets the top of form counter.

On some printers, the ESCAPE CHR\$(33) sequence is interpreted as a *change to PC emulation mode*. If your printer has this capability, you must cycle the printer power after issuing the ESCAPE sequence.

- GRAPHICS.COM does not operate properly to print color graphics on the Tandy CGP-220 printer unless LPDRVR is disabled. LPDRVR must also be disabled to download character fonts or for using the high-resolution graphics mode on the Tandy 2100P printer.
- LPDRVR detects graphical data sequences and passes these codes directly to DMP printers. When graphics printing is completed, the internal parameters are set to top of form. The printer must be manually set to top of form.

MLPART.SYS

(For Multiple DOS Partitions)

MLPART.SYS is an installable device driver that lets your computer access multiple, non-bootable (DOS2) partitions on a hard disk drive. To create these partitions, you use the MLPART.COM command. (See “MLPART” in Part 2 of this manual.)

The purpose of MLPART.SYS is to let you take advantage of all the space on a hard disk that has 32 or more megabytes of storage. The bootable (DOS) partition must begin and end in the first 32 megabytes of the hard disk.

Note: The hard drive for which you want to use MLPART.SYS is likely to be a non-standard drive. If it is, you need to install HDRIVE.SYS before you can use the drive at all. (See “HDRIVE.SYS,” earlier in this appendix.)

Before installing the device driver, create the DOS2 partitions, using the MLPART command. Then install the device driver by placing the following line in your CONFIG.SYS file:

```
DEVICE=MLPART.SYS drive specifications
```

where *drive specifications* is a list of the physical drive letters of the hard disks containing DOS partitions.

For example, if you have only one DOS2 partition on Drive C, you use the line:

```
DEVICE=MLPART.SYS C:
```

If you have three DOS2 partitions on Drive C and one on Drive D, you can use:

```
DEVICE=MLPART.SYS C: C: D: C:
```

MLPART assigns each DOS2 partition a logical drive letter, and tells you the letters. In doing this, it adheres to the order in which you created the DOS2 partitions. Therefore, in the previous example, if no other block devices exist, the DOS2 logical drives are E, F, and H (on Drive C) and G (on Drive D).

Install the device driver only once. After you do so, use MLFORMAT to format the partitions according to their logical drive letters.

MODEVM.SYS

(For International Use of the VM-1)

MODEVM is an installable device driver that improves video display output on a VM-1 monitor (Cat. No. 26-5111) used outside the U.S.

To install this driver, international users need to place the following command in the CONFIG.SYS file:

```
DEVICE=MODEVM.SYS
```

SPOOLER.SYS

(For Buffering Data to the Printer)

SPOOLER.SYS is a loadable driver that buffers data to the printer by temporarily placing the data in memory. SPOOLER *spools* the data to the printer whenever the printer is available. In this way, you can continue to use your computer to process data while printing other data.

The DEVICE command that you need to place in CONFIG.SYS to install the printer buffer driver is:

```
DEVICE=SPOOLER [printer] [size] [/E]
```

where:

- | | |
|----------------|--|
| <i>printer</i> | is the number of the printer you want to use. It can be either 1 or 2. If you omit the number, the system assumes you want to use Printer 1. Notice that you must precede the number with a slash (/). You can install two spoolers if you have two printers connected. |
| <i>size</i> | is the size of the buffer in kilobytes. If you omit <i>size</i> , the system uses 16K. |
| /E | stores the data in expansion memory (an area of memory above the 1 megabyte boundary on the main board). You can use this option only if you have the VDISK.SYS device driver installed on your Tandy 3000. (You cannot use /E with a Tandy 3000 HL computer.) See "VDISK.SYS" for more information. |

VDISK.SYS

(For the Virtual Disk)

Accessing information from floppy or hard disk is slow compared to accessing information from your computer's memory. This is because of the time it takes for the disk to get up to speed and for the disk's read/write head to locate the data. Using VDISK, you can set aside portions of your computers *random access memory* (RAM) that simulate disk storage. These areas are called virtual disks.

The features of VDISK are:

- When you create a virtual disk, the system automatically assigns it a drive name. If you have only two disk drives—floppy drives A and B—and you install a virtual disk, the system gives the virtual disk the name C.
- Virtual disks can occupy extended memory if your computer is so equipped.
- You can assign volume labels to virtual disks.
- You can select the amount of memory for each virtual disk to use.

Warning: If you turn off or reset the computer, you lose the contents of virtual disks. Copy the data to floppy or hard disk before turning off or resetting your computer.

The `DEVICE` command that you need to place in `CONFIG.SYS` to install the virtual disk driver is:

```
DEVICE=VDISK.SYS [storage sector-size dir /E]
```

storage specifies the size (in kilobytes) that you want the virtual disk to be. It can be in the range of 1 to the capacity of your computer's memory. The default value is 64K.

If the installation of `VDISK` leaves fewer than 64 kilobytes free, `VDISK` reduces the amount of storage it sets aside.

sector-size is the sector size (in bytes) of the virtual disk's format. You can select 15, 256, or 512 bytes. If you omit *sector-size*, or if the size you specify is not valid, `VDISK` sets the size to 128.

dir specifies the number of entries allowed in a virtual disk's directory. You can select a value in the range 2 to 512. The default value is 64. `VDISK` might adjust the *dir* value up to the nearest sector size boundary. If the virtual disk is too small to hold the FAT (*file allocation table*), the directory, and at least two sectors, `VDISK` reduces the directory size. The volume label resides in one of the directory entries.

/E specifies that `VDISK` is to reside in *expansion memory* (an area of memory above the 1 megabyte boundary on the main board), if your computer is so equipped. (The */E* switch is available for the Tandy 3000 only.)

INDEX

.com conversion of executable file 86
/CR 115
/LF 115
3½-inch diskette, formatting 105
3½-inch drive, configuring in system 372
5¼-inch, double-sided diskette
 formatting in a high-capacity drive 105, 108, 109
 formatting in a standard drive 105, 108, 109
5¼-inch, high-density diskette, formatting 105, 108, 109

active partition 95, 96, 133
adding library modules 265
adding lines to a file in memory 205
align types 257
ampersand on library manager command line 266
APPEND 41-42
application program 347
 returning control to application 8
application programs, using in different drives 43
archive attribute 45
 controlling selective backups 45
 displaying 45
 setting and clearing 45
argument 347
ASCII 347
ASCII and scan codes 353-363
ASCII characters 112, 128-255
ASCII file 61
assemble 347
Assembler 347
ASSIGN 43-44
asterisk on library manager command line 265
ATTRIB 45
attributes, displaying for all files 45
AUTOEXEC.BAT 25-26, 161
AUX 11, 58, 67

background printing 145-146
BACKUP 43, 45, 46-48
 placement of backup files 46
backup file, created by EDLIN 193
backup log entry 47

BACKUP/RESTORE compatibility 47, 157
batch file 347

- conditional execution of commands 120-121
- creating 23-24
- displaying message 141-143
- executing 24
- including remarks 24, 152
- pausing 24
- percent sign 29
- protecting information in file 81
- suspending execution 141-143
- transferring control to another line 110-111
- where to create 25
 - calling one from another 29
 - replaceable parameters 26-28

batch file process, summary 25
battery, replacing 165
baud rates 135
binary 347
binary file 61, 347

- displaying 180

black-and-white printer 115
block device, defining parameters 371-372
bootable disk 107
BREAK 49
buffer 348
byte alignment 257
byte-by-byte file comparison 92

C language 246
canonical frame number 257
carriage return 4, 114
case, ignoring when comparing files 89
case-sensitive languages 247
CGP-220 printer 113, 115
character scan codes, converting to US scan codes 124-125
character

- copying 4
- deleting 5
- displaying 3, 4
- inserting 5

CHDIR 50-52
CHKDSK 53-56
class 255
CLS 57

- CMOS date and time, changing 165
- CMOS, initializing system configuration parameters 165-166
- COM1 11, 58, 67
- COM2 11, 58, 67
- combine type 258-259
- comma on linker command line 235
- command 348
 - editing 187-190
 - entering 1
 - executing for several items 103-104
 - maximum number of characters 1
 - processing 4
- COMMAND 58-59
- command path, setting or displaying 139-140
- command piping 22
- command processor 376
 - specifying a string to execute 59
 - starting a new processor 58-59
 - telling processor not to exit to a higher level 58
- command types 31
- COMMAND.COM 58
 - copying 176
- commands *see* CONFIG.SYS commands, debug commands, EDLIN commands, linker commands, MS-DOS commands
- commas used in linker command line 235
- common combine type 258
- CON 11, 58, 67, 348
- CONFIG.SYS 59, 105, 161, 175
 - creating 365
 - location 365
 - sample file 377
- CONFIG.SYS commands 366
 - BREAK 367
 - BUFFERS 368
 - COUNTRY 369
 - DEVICE 370
 - DRIVPARM 371-372
 - FCBS 373
 - FILES 374
 - LASTDRIVE 375
 - SHELL 376
- console 58, 67
- continuous timeout retries on RS232 port 135
- control characters, inserting into text with EDLIN 202
- control key 3

control keys *see* special keys
control, returning from MS-DOS to previous level 87
COPY 60-66
 appending files 66
 combining files 64-65
 copying files 60-63
 default values for /A and /B on a straight copy 61
 default values for /A and /B when appending 64
 default values for /A and /B when combining 66
copying lines with EDLIN 206-208
country code, selecting 160-162
CPU speed, changing 135
creating smaller executable files 245
CTTY 67
current directory 16-17, 348
 changing 17, 50-52
 changing to a directory on another disk 51
 displaying 50-52
current drive 15, 348
 changing 15
cursor 3, 348
cylinders 38, 118

data file path, displaying 41
data file *see* ASCII file
data files, setting search path 41-42
data sorting 171-172
databits 135
DATE 68-70
date format, changing 69
date 68-69
DEBUG command parameters 271-273
DEBUG commands 270
 ASSEMBLE 275-277
 COMPARE 278
 DUMP 279-280
 ENTER 281-282
 FILL 283
 GO 284-285
 HEX 286
 INPUT 287
 LOAD 288
 MOVE 290-291
 NAME 292-293
 OUTPUT 294

- PROCEED 295
- QUIT 296
- REGISTER 297-299
- SEARCH 300
- TRACE 301-302
- UNASSEMBLE 303-304
- WRITE 305-306
- DEBUG, starting 267-268
- default 35, 348
- defective tracks 118
- DEL *see* ERASE 71
- deleting
 - from template 190
 - library modules 265
 - lines with EDLIN 209-211
 - strings with EDLIN 224
- delimiter 36, 348
- Deluxe Graphics Adapter 112
- destination 36
- device 348
- device driver *see* installable device drivers
- device errors 345-346
- device names 11
- DIR 72-74
- directory 5, 348
 - backing up *see* BACKUP
 - changing 50-52
 - checking 53-56
 - copying 184-186
 - creating 14, 129
 - deleting 14
 - current *see* current directory
 - displaying 12, 72-74
 - fixing errors in a directory 53
 - parent *see* parent directory 18
 - removing 158-159
 - ROOT *see* ROOT directory 7
 - wide display 72
 - wide mode 74
- directory names 7
 - anonymous 18-19
- disk buffers, setting number 368
- disk errors 345-346
- disk file 349
- DISKCOMP 75-77

- requirements of target diskette 76
- DISKCOPY 43, 78-79
 - requirements of target diskette 79
 - single-drive copy 78
- diskette copying 78-79
- diskette formatting
 - 8 sectors per track 106
 - 8 sectors per track and room for system 107
 - standard diskette for 720K 101-102
- diskettes, comparing 75-77
- diskette types 105
- disks, backing up *see* BACKUP
- DISKTYPE 80
- display 349
- displaying a command line 188-189
- displaying lines with EDLIN 218-220
- DMP 134
- DMP-110 printer 113
- DOS2 partitions
 - accessing 131, 391
 - creating 131-133
 - formatting 130
- double-sided diskette, formatting as single-sided 107
- drive 349
 - backing up *see* BACKUP
 - current *see* current drive
 - joining to a pathname 122-123
- drive designation, setting last valid designation 375
- drive letter, reassigning 43-44
- drive size and capacity 80
- drives, support of additional drives 382
- drive type 80
- dummy argument 349
- dump 349
- DWP 134

- ECHO 81-82
- editing a file in parts 205, 230
- editing functions 2
- editing keys 1-2 *see also* EDLIN editing keys
- EDLIN command format and rules 201-203
- EDLIN command parameters 203-204
- EDLIN commands 201-230
 - Append Lines 205
 - Copy Lines 206-208

- Delete Lines 209-211
- Edit Line 212-213
- End Edit 214
- Insert 215-217
- List 218-220
- Move Lines 221
- Page 222
- Quit 223
- Replace String 224-225
- Search Text 226-228
- Transfer Lines 229
- Write Lines 230
- EDLIN command summary 201
- EDLIN editing keys
 - copy all 195
 - copy character 194
 - copy to character 194
 - delete character 195
 - delete to character 196
 - enter line 199
 - insert 197
 - replace template 198
 - void line 196-197
- empty subdirectories, copying 184-186
- end line 4
- end-of-line character 114
- entering multiple commands *see* batch files, command piping
- environment settings, displaying 163
- environment size 58
- environment strings, setting one equal to another 163-164
- EOF character 61, 62, 63
- ERASE 83-84
- error 349
- ERRORLEVEL 120
- errors 307-346
- errors CHKDSK cannot correct 55
- errors CHKDSK corrects 54
- EXE2BIN 85-86
- executable file 233
 - converting to binary file format 85-86
 - default filename 234
- execution, stopping 3
- EXIST 120-121
- EXIT 87
- expansion memory 166, 393, 395

- extended printer capabilities 386
- extended screen and keyboard features 381
- extending lines in linker command 233
- extension 8-9, 349
- external command 31, 349
 - order of search 139
 - transferring to another disk 62
- external references 231
- extracting library modules 265

- FC 88-93
- FDISK 94-97, 117, 131
 - menu 95
- File Allocation Table 150
- file comparison *see* FC
- file control blocks 373
- file editing 191-199
 - with EDLIN 193
- file fragmentation 79
 - reducing 62
- file handles 374
- file sharing 167
 - allowing over a network 45
 - backing up shared files 47
 - system 7
- filename 7, 8, 349
 - examples 9
- filename extension *see* extension
- files
 - adding to backup 48
 - appending to an existing file 66
 - ASCII comparison 88, 89
 - binary comparison 88
 - checking 53
 - combining 64-65
 - comparing 88-93
 - copying 60-63, 184-186
 - copying to disk of a different format 184-186
 - creating 12-13
 - creating with EDLIN 191-192
 - deleting 83-84
 - displaying 13, 179, 180
 - displaying date and time modified 72
 - finding 67
 - making minor changes 138

- merging with EDLIN 229
- packing on backup diskette 48
- paging through with EDLIN 222
- patching 138
- protecting from closure 373
- recovering 150-151
- renaming 153
- restoring backed up files 156-157
- saving a file with EDLIN 192
- updating 154-155
- verifying 182
- filespec 349
- filter 22, 349
- FIND 22, 98-100
- fixups 259-260
- floppy diskette, formatting *see* FORMAT
- FMAT2000 101-102
- FOR 103-104
- format 350
- FORMAT 43, 94, 105-109, 117, 131
- Format Hard Disk 94, 106
- FORTRAN 246
- function keys 187-188
 - copy all 188
 - copy character 187
 - copy to character 187
 - delete character 187
 - delete to character 187
 - end-of-file 188
 - enter line 188
 - insert 188
 - replace template 188
 - void line 188
- glossary 347-352
- GOTO 110-111
- GRAFTABL 112
- GRAPHICS 113-116
 - menu 115
- graphics screen, reproducing on printer 113-116
- GRAPHICS, disabling printer driver with CGP-220 printer 390
- greater-than symbol 21
- group 255, 259
- group address 255
- group information in map file 241

- hard disk
 - fewer than 32 megabytes *see* FDISK
 - formatting entire disk for MS-DOS 95, 97
 - formatting for multiple operating systems 94
 - formatting track and sector information *see* HSECT
 - more than 32 megabytes *see* MLPART
- hard disk drive type 118, 165, 384
- hard disk heads, parking 170
- hard disk initialization 94
- hard disk partition
 - accessing DOS2 partitions *see* MLPART.SYS
 - active 95
 - non-active 96
 - maximum number of partitions 95
 - multiple partitions 96
 - status 96
 - changing *see* FDISK and MLPART
 - changing size 96
 - creating *see* FDISK and MLPART
 - displaying information 97
 - formatting *see* FORMAT and MLFORMAT
- hard disk partitioning, selecting next disk 97
- hardware configuration, changing 165-166
- heads 38, 118
- high-capacity floppy disk drive 101
- home directory 17
 - changing 50-52
- HSECT 94, 106, 117-119, 131
- IF 120-121
- input 350
- input/output, redirecting 21
- input/output device, changing 67
- inserting characters in a command line 189
- inserting text with EDLIN 215-217
- installable device drivers 370, 379-380
 - ANSI.SYS 381
 - DRIVER.SYS 382-383
 - HDRIVE.SYS 384-385
 - LPDRVR.SYS 386-390
 - MLPART.SYS 391
 - MODEVM.SYS 392
 - SPOOLER.SYS 393
 - VDISK.SYS 394-395
- interleave factor 118

- internal 350
- internal commands 31
- internal line buffer 89
- internationally configured MS-DOS diskette 161-162
- JOIN 122-123
- keyboard device 11
- keyboard layout code, selecting 160-162
- keyboard program, replacing with international program 124-125
- KEYBxx 124-125
- keys
 - control-break 3
 - control-c 3, 37, 77
 - control-h 3
 - control-j 4
 - control-numlock 3
 - control-p 4
 - control-print screen 4
 - control-s 3
 - delete 5
 - enter 4
 - f1 4
 - f2 4
 - f3 4
 - insert 5
 - left arrow 5
 - right arrow 3
- keywords 37
 - synonymous 39
- LABEL 126-127
- LASTDRIVE 175
- less-than symbol 21
- LF 128
- library files 231
- library manager 251-256
 - command characters 265-266
 - map file 251
 - order of operations 251
 - starting with command line 263-264
 - starting with prompt method 262-263
 - starting with response file 264-265
- library modules, modifying 251

- library search 240
 - order 240
- line feed 114
 - suppressing line feed after carriage return 128
 - turning on and off 136
- line printer 11
- line
 - displaying 4
 - specifying in EDLIN commands 203
- linker operation, parts 232
- linker options 253
 - allocating a data group 250
 - copying line numbers to map file 246-247
 - ignoring default libraries 248
 - packing executable files 245
 - pausing to change disks 244-245
 - preserving lowercase 247
 - producing a public-symbol map 246
 - removing groups from a program 251
 - setting high start address 250
 - setting maximum allocation space 249
 - setting maximum number of segments 252
 - setting stack size 248-249
 - setting the overlay interrupt 251-252
 - using DOS segment order 253
- linker starting 233-239
 - command line method 235-237
 - prompt method 233-235
 - response file method 238-239
- linker temporary file 243
- listing file *see* map file 231
- loading a line for editing with EDLIN 212-213
- logical drive letter 130
- long reference 260
- LPT 11, 350

- machine language code 231
- Macro Assembler 246
- map file
 - library 263
 - linker 231, 241-242
 - pausing before creation 245
- MCOPY.EXE 185
- Media Error Map 118
- media types

- backing up between different types *see* BACKUP
- drives to use 107
- FORMAT switches 108
- memory combine type 259
- merging files with EDLIN 229
- messages 307-346
- messages, displaying during batch file execution 81-82
- minus sign on library manager command line 265
- mistakes, correcting 1-3
- MKDIR 14, 129
- MLFORMAT 130, 131-132
- MLPART 94, 131-133
 - menu 131-132
- MLPART.COM 131
- MLPART.SYS 131, 391
- MODE 128, 134-136
- modified files
 - backing up 45, 46, 48
 - copying 184-186
 - restoring 156-157
- MORE 22, 92, 137
- moving lines with EDLIN 221
- MS-DOS commands *see* individual command name
- MS-DOS command summary 32-35
- MS-DOS version, displaying 181
- multiple DOS partitions, accessing 391
- near segment-relative reference 260
- near self-relative reference 260
- networking 167
- NL 134
- non-active hard disk partition 96, 133
- non-bootable partitions 38
- non-DOS hard disk partitions 96
- non-standard hard disk 118
 - support 384-385
- NUL 11
- output 350
- output, sending to printer 4
- overlay modules 237
- overtyping in a command line 189
- page alignment 257
- page mode 72

- paging through a file with EDLIN 222
- paragraph alignment 257
- parameter 35-36, 350
- parent directory 18
- parity 135
- partition data, displaying 133
- partition size 132
- partitioning
 - selecting next drive 133
 - synopsis of procedure 131-132
- partitions *see* hard disk partitions
- Pascal 246
- PATCH 138
- path 350
- PATH 139-140
- pathname 10-11, 350
 - substituting a virtual drive name for a pathname 175
- PAUSE 24, 141-144
- period
 - for current directory 18
 - for current line 203
- periods, parent directory 18
- peripheral 350
- pipng 350
- plus sign on library manager command line 265
- pound sign to indicate last line of program 203
- processor 376
- PRINT 43, 145-146
- print queue 145-146
- print spooler 173-174
- printer compatibility 113-114
- printer control codes 388-389
- printer data, buffering 393
- printer driver 386-390
 - disabling 390
- printer parameters 136
- printer type 113
 - setting 134
- private combine type 259
- PRN 11, 350
- program 350
 - executing 1
- PROMPT 147-149
- public combine type 258
- public symbol addresses 242

- public symbols below the start of a program 242
- pure binary conversion of executable file 85-86

- quitting EDLIN 223

- read-only attribute 45
- read-only files
 - restoring 156
 - updating 154
- read/write reliability 107
- RECOVER 150-151
- relative line numbers in EDLIN 202
- REM 24, 152
- REN 153
- REPLACE 154-155
- replaceable parameters 350
 - more than 10 168-169
- replacing strings with EDLIN 224-225
- response file, ending 239
- RESTORE 45, 156-157
 - finding which disk to restore 47
- RMDIR 158-159
- ROOT directory 7, 351
 - creation 8
 - joining to a different pathname 122-123
- RS232 communication parameters, setting 135, 136
- RS232 ports 11, 58, 67
- run file 233

- saving a file after editing with EDLIN 214
- saving in the template 190
- screen print, translating video characters properly 134
- screen scroll 3
- screen
 - clearing 57
 - displaying one screen at a time 22, 136-137
 - printing contents of screen 4
 - shifting left and right 134, 136
- searching for strings with EDLIN 225-228
- sector 150-151
- sectors, verifying after copy 61
- segment 255
- segment combine type 258-259
- segment information in map file 241
- segments, order copied to executable file 256, 258

SELECT 160-162
 semicolon
 library manager command line 266
 linker command line 235, 236
SET 163-164
SETUP 165-166
SHARE 167
SHIFT 27, 168-169
SHIPTRAK 170
 short reference 260
SORT 22, 171-172
source 36
source code 231
space bar 3
special keys 1-5
special type 37
SPOOLER 173-174
stack combine type 258
standard, double-sided diskette
 formatting in a high-capacity drive 105, 108, 109
 formatting in a standard drive 105, 108, 109
stopbits 135
string 204, 351
string, searching for with EDLIN 225-228
strings
 deleting with EDLIN 224
 replacing with EDLIN 224-225
subdirectories
 copying 184-186
 displaying 179
 restoring 156
 uses 129
SUBST 175
switch 351
switches 36
syntax 37, 351
SYS 176
system 351
system base memory 166
system configuration parameters, initializing 165-166
system disk, creating and updating 176
system prompt 351
 changing 147-149

Tandy 2000-compatible diskettes 102

target 36
template 187-188, 351
 displaying 3
terminate line 4
text file *see* ASCII file
text
 finding 22, 98-100
 sorting 22
TIME 177-178
time
 changing format 178
 setting or displaying 177-178
toggle 351
track 351
TREE 67, 179
TTY 67
TYPE 180

updating files 154-155
US scan codes 161-162

VER 181
VERIFY 182
verify 352
verifying sectors after copy 61
video adapter type 166
video mode 134
video monitor, setting characters-per-line 134, 136
virtual disk 394-395
virtual drive name 175
VM-1 monitor 392
VM.TMP file 243
VOL 183
volume label 38, 101, 107
 creating, changing, deleting 126-127
 displaying 73, 183
 length 126
 required for subsequent formatting 108

wild card 10, 352
 renaming files 153
word alignment 257
writing edited lines to disk 230

XCOPY 45, 184-186

RADIO SHACK, A Division of Tandy Corporation

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

AUSTRALIA	BELGIUM	FRANCE	U. K.
91 Kurrajong Avenue Mount Druitt, N.S.W. 2770	Rue des Pieds d'Alouette, 39 5140 Naninne (Namur)	BP 147-95022 Cergy Pontoise Cedex	Bilston Road Wednesbury West Midlands WS10 7JN