

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 1000

MS-DOS
REFERENCE
MANUAL

Tandy 1000 EX/SX BIOS Compatibility Software: © 1984, 1985, 1986 Tandy Corporation and Phoenix Software Associates, Ltd. All Rights Reserved.

MS-DOS® Vers. 2.11 Software: © 1981, 1982, 1983 Microsoft Corporation. Licensed to Tandy Corporation. All Rights Reserved.

MS-DOS® Vers. 3.2 Software: © 1981, 1986 Microsoft Corporation. Licensed to Tandy Corporation. All Rights Reserved.

GW™-BASIC Vers. 2.02 Software: © 1982, 1984 Microsoft Corporation. Licensed to Tandy Corporation. All Rights Reserved.

GW™-BASIC Vers. 3.20 Software: © 1983, 1984, 1985, 1986 Microsoft Corporation. Licensed to Tandy Corporation. All Rights Reserved.

All portions of this software are copyrighted and are the proprietary and trade secret information of Tandy Corporation and/or its licensor. Use, reproduction, or publication of any portion of this material without the prior written authorization of Tandy Corporation is strictly prohibited.

Tandy 1000 MS-DOS Reference Manual:

© 1986 Tandy Corporation.

All Rights Reserved.

Reproduction or use, without express written permission from Tandy Corporation and/or its licensor, of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors in or omissions from this manual, or from the use of the information contained herein.

Tandy and Radio Shack are registered trademarks of Tandy Corporation.

Microsoft, MS, and MS-DOS are registered trademarks of Microsoft Corporation.

GW is a trademark of Microsoft Corporation.

IBM is a registered trademark of International Business Machines Corporation.

ABOUT THIS MANUAL

MS-DOS® is an extremely powerful computer managing tool and has numerous commands and options. If you are a beginner, don't let this worry you. MS-DOS is not difficult to use.

This section provides information that greatly facilitates your use of the MS-DOS commands listed in Part 1. **Be sure to read it before going on to Part 2.**

This manual helps you make immediate use of MS-DOS's functions. Part 1 describes how to begin using the operating system's commands and editing features as well as the organization of directories and files.

Part 2 begins by giving you specific instructions on how to interpret and use command references. The command references which follow provide the syntax, parameters, special instructions, and examples of use for each of the MS-DOS commands.

The MS-DOS powerful line editor, EDLIN, is described in detail, and with numerous examples, in Part 3. Your understanding of the special features in this section will greatly increase the advantages of MS-DOS and save you time.

Part 4 is for assembly language programmers. It describes the functions of the MS-DOS Linker. LINK is used to produce *machine language* code from assembler or compiler object modules.

The MS-DOS DEBUG program provides for the testing of executable object files and lets you alter the contents of a file or register. The use of DEBUG functions and commands is described in Part 5.

The use of MS-DOS is greatly facilitated by its descriptive error messages. The error messages you might encounter when using any of the MS-DOS functions are listed in Part 6.

The *CONFIG.SYS* file, described in Appendix B, lets MS-DOS install device-specific drivers as part of the system start-up. With *CONFIG.SYS* you can establish the number of buffers and files, install a printer driver and set other parameters as part of the system boot process.

CONTENTS

Part 1. Introduction

1	How to Use MS-DOS	1
	Entering a Command	1
	Executing a Program	1
	Editing and Special Keys	1
	The Control Keys	3
2	Organization of Information	5
	The File System	5
	The Root Directory	6
	Filenames	6
	Extensions	6
	Examples of Filenames	7
	Wild Cards	8
	Pathnames	8
	Device Names	9
	Directories	9
	Creating Directories	12
	Deleting Directories	13
	The Current Drive	14
	The Current Directory	14
	Home Directories	16
	Anonymous Directory Names	17
3	Redirecting Commands	19
	Input And Output	19
	Filters	20
	Command Piping	20
4	Batch Files	21
	Executing Several Commands	21
	The Batch File	21
	Executing a Batch File	22
	Summary of the Batch File Process	23
	The Autoexec.bat File	24
	Batch Files With Replaceable Parameters	24
	Creating a Batch File With Replaceable Parameters	25
	Executing a Batch File With Replaceable Parameters	25
	Sample Use of Replaceable Parameters	26
	Reminders about Batch Files	27

Part 2. MS-DOS Commands

5 MS-DOS Commands	29
Quick Reference To MS-DOS Commands	30
How to Use the Command Reference	33
Command Structure and Syntax	34
Organization of MS-DOS Commands	35
The Use of Special Type	35
Synonymous Keywords	36
6 Command Reference	37

Part 3. EDLIN

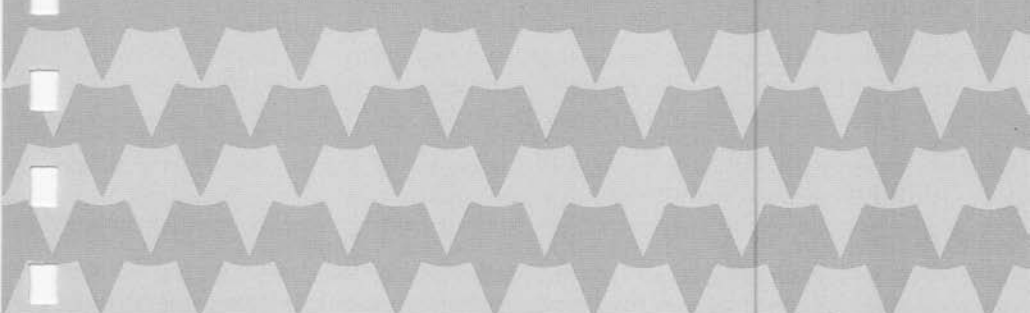
7 The Line Editor (EDLIN)	189
Function Keys	189
8 Editing Commands	191
The Template	191
Display and Change	191
Overtyping and Insert	192
Saving in the Template	192
Deleting	193
9 Editing Files	195
Creating a File	195
Inserting Text	196
Saving Your Files	196
Editing an Existing File	197
Using the Special EDLIN Editing Keys	197
10 EDLIN Command Reference	205
Command Format and Rules	205
Command Summary	207
Command Parameters	207

Part 4. MS-LINK

11 Linking Object Modules	235
Basic Information	235
System Requirements	235
Linking Object Modules and Producing a Run File	236
Resolving External References	236
Producing a List File	236
The VM.TMP (Temporary) File	237
Definitions	237
Command Prompts	239

Command Characters	241
Linker Switches	242
Starting the Linker	244
Sample Link Session	246
12 LINK Technical Reference	249
Definitions	249
How the Linker Combines and Arranges Segments	250
How the Linker Assigns Addresses	253
Relocation Fixups	254
MS-LIB The Library Manager	256
Order of Operations	256
Running MS-LIB	257
Command Characters	260
Part 5. DEBUG	
13 The DEBUG Utility	263
DEBUG Commands	264
Command Summary	265
Command Parameters	265
Part 6. Messages	
14 Problems and Error Messages	293
MS-DOS Error Messages	293
Device Error Messages	293
Command Error Messages	297
EDLIN Error Messages	305
Linker Error Messages	308
DEBUG Error Messages	310
Part 7. Appendices	
A ASCII and Scan Codes	311
B Configuring Your System	321
C Installable Device Drivers	331
D Hard Disk Setup	343
Glossary	345
Index	351

Part 1
Introduction



HOW TO USE MS-DOS

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.

Entering a Command

You can enter a command whenever the screen displays the system prompt. The command may have a maximum of 127 characters, including any combination of uppercase or lowercase letters. End each command by pressing **ENTER**. For example, type:

```
cl s ENTER
```

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

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 due to one error, MS-DOS provides several special editing functions.

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

Before entering a command. If you catch a mistake before you have pressed the **ENTER** key, you can use one of the following keys to correct the mistake:

- **BACKSPACE** — 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 the command to be executed. After execution, or if you made an error, you have the following options:

- Reexecute the same command. To do so, type **F3** **ENTER**.
- Edit the line and reexecute 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. Some functions you can perform are:

- 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

These editing features are also available in the MS-DOS file editing program, EDLIN. EDLIN also offers these features:

- 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 editing functions, MS-DOS provides a number of control keys. A control key affects a command or command line. The keys are listed below.

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

Keys

Function

CTRL **C**

stops execution of a command.

SHIFT **PRINT**

sends everything currently displayed on the screen to the line printer.

PRINT, **CTRL** **N**,
or **CTRL** **P**

sends all output to the line printer, as well as to the screen. Press again to stop the function.

BACKSPACE or
CTRL **H**

removes the last character from the command line and erases the character from the screen.

CTRL **J**

causes a line end and carriage return but does not send the line to be executed or processed. You can continue to add to the command line before pressing **ENTER** to terminate input.

HOLD or
CTRL **S**

halts scrolling. Press **HOLD** or **CTRL** **Q** to resume scrolling.

ESC

voids the current line and clears the command line buffer. It then outputs a backslash (\), carriage return, and line feed. The *template*, a storage area that contains the last MS-DOS command you used, is not affected. Although the system prompt is not displayed, the system is ready for a command.

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.

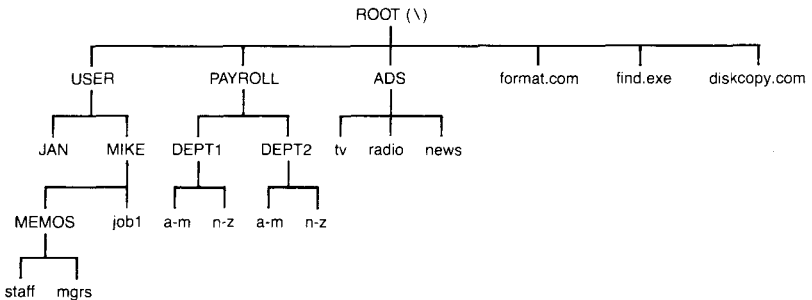
The File System

Groups of files can be collected into *directories*, much as you would use a file cabinet drawer to collect all the file folders pertaining to a particular subject. These directories, in turn, can be collected 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.



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

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 (\). (Type 7 on the numeric keypad with the `NUM LOCK` key off.)

In the sample disk diagram on the previous page, three directories (USER, PAYROLL, and ADS) have been created directly in the root directory. (This root directory also contains the `Format.com`, `Find.exe`, and `Diskcopy.com` files.) Further, the USER and PAYROLL directories each contain subdirectories (MIKE, DEPT1, and DEPT2). Finally, the MIKE subdirectory contains a MEMOS subdirectory. All but one of the directories on the sample disk contain files.

Filenames

Each file has a name that can be a maximum of eight characters long. You can use letters, numbers and some symbols. 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 (.); they 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 file type, such as the following:

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

If you add the extension *.dat* to a file named *invntory*, the file-spec becomes:

```
invntory.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
PROG1.bas
SAMFILE
2AR.dat
```

The examples below are illegal filenames:

<code>max*min</code>	* isn't a legal character for names
<code>.DATA</code>	the period can be used only to separate the filename and extension
<code>open ordr</code>	a name cannot contain a space

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.

Wild card examples and the results you can expect are given below.

You type:	MS-DOS finds:	Examples:
test?run.exe	All files in one directory that begin with <i>test</i> , have any character next, followed by <i>run</i> with the extension <i>.exe</i> .	test1run.exe test3run.exe
test*.exe	The files above plus all files in one directory that begin with <i>test</i> and have the extension <i>.exe</i> .	test.exe testall.exe test1.exe
oldfile.*	All files named <i>oldfile</i> (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 or *pathnames* to find files. A pathname is a list of directories and filenames from the root directory down to the file you want to access. Each pathname can have a maximum of 63 characters.

Refer to the previous MS-DOS disk diagram. To access the Radio file, tell MS-DOS the path to follow from the root directory to the file, separating the *branches* with backslashes. For instance, to display the contents of the Radio file, type:

```
type \ads\radio 
```

MS-DOS reads the pathname from left to right to determine that the file you want is in the ADS directory and that its name is Radio. It then lists the contents of Radio 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.

Device Names

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

- **AUX** (auxiliary) usually refers to the RS-232 serial port but can be any device you specify using the CTTY command.
- **CON** (console) refers to the screen or keyboard.
- **PRN** (printer) refers to Printer 1.
- **LPT1, LPT2, LPT3** (line printer) refer to Printer 1, Printer 2, and Printer 3, respectively.
- **COM1, COM2** (communications) refer to serial Port 1 and Port 2.
- **NUL** (null) refers to a non-existent device.

Note: You cannot use any of the device names above as file names.

Directories

MS-DOS directories are collections of files. Look at the following example to understand how they are created and used.

Assume that the disk in Drive B is newly formatted and contains only a root directory. You can use EDLIN to create the test file on Drive B. To do so, type:

```
edlin b:\file1.tst 
```

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 [ENTER]
```

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 [F6] [ENTER] or [CTRL] [Z] [ENTER] to end the file. You do not type line numbers; they are entered automatically by EDLIN.

To create a file, type:

```
1:*This is my test file.[ENTER]
2:*It shows the use of directories.[ENTER]
3:[F6] [ENTER]
```

After you type [F6] [ENTER], EDLIN displays the asterisk to indicate it's ready for another command. To exit EDLIN and return to MS-DOS, type:

```
e [ENTER]
```

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

```
dir b:\ [ENTER]
```

The listing now indicates the existence of the new file:

```
Volume in drive B has no label
Directory of B:\
FILE1 TST  58    8-24-86    9:07a
1 File(s) nnnnnn bytes free
```

The number of free bytes available in your computer's memory is represented as *nnnnnn*. Your directory listings may differ from those shown in this manual.

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

```
type b:\file1.tst [ENTER]
```

The screen shows:

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

Use EDLIN to create two more text files.

```
edlin b:\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 b:\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 b:\ 
```

The screen shows:

```
Volume in drive B has no label
Directory of B:\

FILE1 TST  58      8-24-86    9:07a
FILE2 TST  60      8-24-86    9:09a
FILE3 TST  59      8-24-86    9:09a

   3 File(s) nnnnnn bytes free
```

Creating Directories

The MKDIR command creates a new directory on any drive. To create a new directory called MYDIR in the Drive B root directory, type:

```
mkdir b:\mydir 
```

MS-DOS automatically makes MYDIR a subdirectory of the Drive B root directory. You can check it by using DIR:

```
dir b:\ 
```

The screen shows:

```
Volume in drive B has no label
Directory of B:\

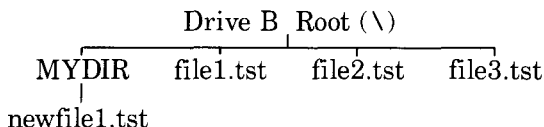
FILE1 TST  58      8-24-86    9:07a
FILE2 TST  60      8-24-86    9:09a
FILE3 TST  59      8-24-86    9:09a
MYDIR      <DIR> 8-24-86    9:09a

   4 File(s) nnnnnn bytes free
```

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

```
copy b:\file1.tst b:\mydir\newfile1.tst 
```


Here's what the structure looks like now:



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

```
dir b:\mydir 
```

The screen shows:

```
Volume in drive B has no label
Directory of B:\MYDIR

.                <DIR>   1-01-80    12:33a
..               <DIR>   1-01-80    12:33a
NEWFILE1 TST 58   8-24-86    9:07a

    3 File(s) nnnnnn bytes free

```

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

Note: See “Anonymous Directory Names” in this chapter for an explanation of the . and .. symbols.

Deleting Directories

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

1. Use the DIR command to see what files are in the directory.
2. Use the COPY command to copy any files you may need to another directory.
3. Use the ERASE command to remove all files from the directory.
4. Use the RMDIR command to remove (delete) the directory.

If there are subdirectories in the directory you wish to delete, follow the previously described procedure for deleting these directories.

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; Drive A 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 B:\PAYROLL, specify the drive and the directory name. For example:

```
dir b:\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:

```
b: 
```

The screen shows a new system prompt, B>, to indicate you are now in Drive B.

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 as your current directory.

Knowing about the current directory lets you take even more shortcuts when specifying pathnames. For example, 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 may 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—one that 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:

```
dir job1
```

the complete pathname \USER\MIKE\job1 is implied.

Again, if you are still in \USER\MIKE and you type MEMOS\mgrs, the complete pathname \USER\MIKE\MEMOS\mgrs is implied.

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

```
copy job1 memos\newjob1 
```

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, simply change drives before using the CHDIR command. For example, to change your current directory to B:\PAYROLL, type:

```
b: 
```

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

```
chdir \payroll  or  
cd \payroll
```

Home Directories

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

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

```
type b:staff 
```

MS-DOS assumes you mean B:\PAYROLL\staff. Similarly, if you type:

```
dir b: 
```

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

Note: It does not show a directory listing of the entire disk. For this, you must type:

```
dir b:\ 
```

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

If you type:

```
copy job1 b:newjob1 [ENTER]
```

MS-DOS interprets this as:

```
copy a:\user\mike\job1 b:\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 just 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 highest-level directory in the path).
- The name “..\..” refers to the directory two levels up.

You can use anonymous names in place of pathnames and/or as the first names in pathnames. Some examples:

```
dir . [ENTER]
```

lists filenames in the current directory.

```
erase ..\taxes84 [ENTER]
```

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

Substitute names may 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 b:..\taxes84 [ENTER]
```

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

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

```
b:   
copy a:job1 
```

This will be interpreted as: COPY A:\USER\MIKE\job1 B:\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 line 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 the output of the DIR command 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. Here is an example in which both the input and output are redirected:

```
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 in a file.
- MORE — Causes 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, as described below.

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, divide them with the *pipe separator*, the vertical bar (|). (You produce the vertical bar by pressing 4 on the keypad when the `NUM LOCK` key is off or by pressing `SHIFT` 4 on the keypad when the `NUM LOCK` key is on.) 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. Type:

```
dir | sort ENTER
```

to display 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.

```
dir b: | >b:direc.fil ENTER
```

makes an alphabetically sorted listing of the current directory of Drive B and sends the listing to the file Direc.fil in the home directory on Drive B. MS-DOS creates the file, if necessary.

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.

To create a batch file, you can specify the complete pathname 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 B:   
pause   
chkdsk B: 
```

To save the commands in a file, type `F6` `ENTER` or `CTRL` `Z` `ENTER`. MS-DOS displays `^Z` to indicate you used the control character. Then it displays:

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

At this time, you can 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 that will not affect the operation of the command. The remarks in the *prepdisk.bat* file are to remind you what the file does. Should 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 may continue execution by pressing any key, or you can halt it by pressing `CTRL` `C`.

Executing a Batch File

To execute your batch file, you must be in the directory that contains the file. (See “The Current Directory” in Chapter 2.) Then, enter the filename without the extension. For example, to execute *prepdisk.bat*, type this at the system prompt:

```
prepdisk ENTER
```

Execution proceeds just 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, we recommend that you create all batch files in the root directory of your disk. Later, when you better understand MS-DOS, you may 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:

- To execute a batch file, you must be in the directory that contains the file. (That directory becomes your current directory and that drive becomes your current drive.)
- 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 the PATH command to tell MS-DOS to search for external commands in the directory that contains the commands.
 - Use the COPY command to move your external commands to the directory that contains your batch file.
 - Use the CHDIR command 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 
```

2. Enter the command lines.
3. Save the file by typing either of the following:

```
  or  
  
```

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

```
filename 
```

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 may want to include replaceable (dummy) parameters. In this way, you can use different sets of data when 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 B disk, enter the batch file name and the drive specification to replace %1, as shown here:

```
prepdisk b: 
```

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 the SHIFT command in Part 2 if you wish to specify more than ten dummy parameters.)

These parameters may 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, the parameters %1 and %2 are replaced sequentially by 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 b:prog2 
```

myfile is substituted for %0, A:prog1 for %1, and B:prog2 for %2.

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

```
copy a:prog1.mac b:prog2.mac   
type b: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   
 
```

Execute the batch file by typing:

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

The result is the same as if you had entered 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 and copies the sorted data to the file Sort.dat. Then it 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.
- The commands in the file named *filename.bat* are executed.
- If you press 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.

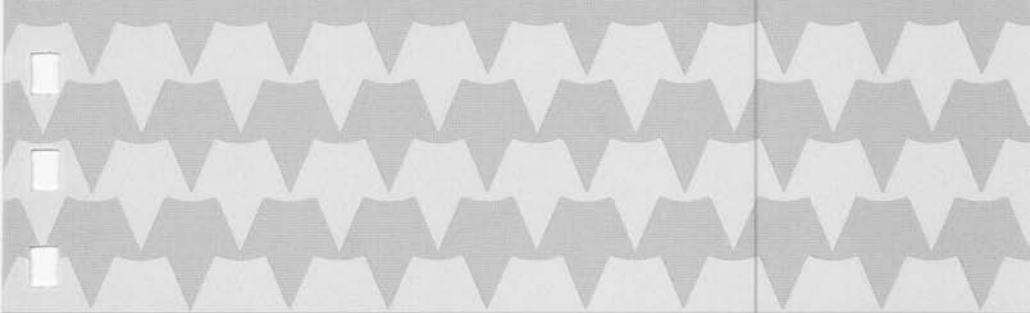
If you press N, the current command ends. Batch processing continues with the next command in the file.

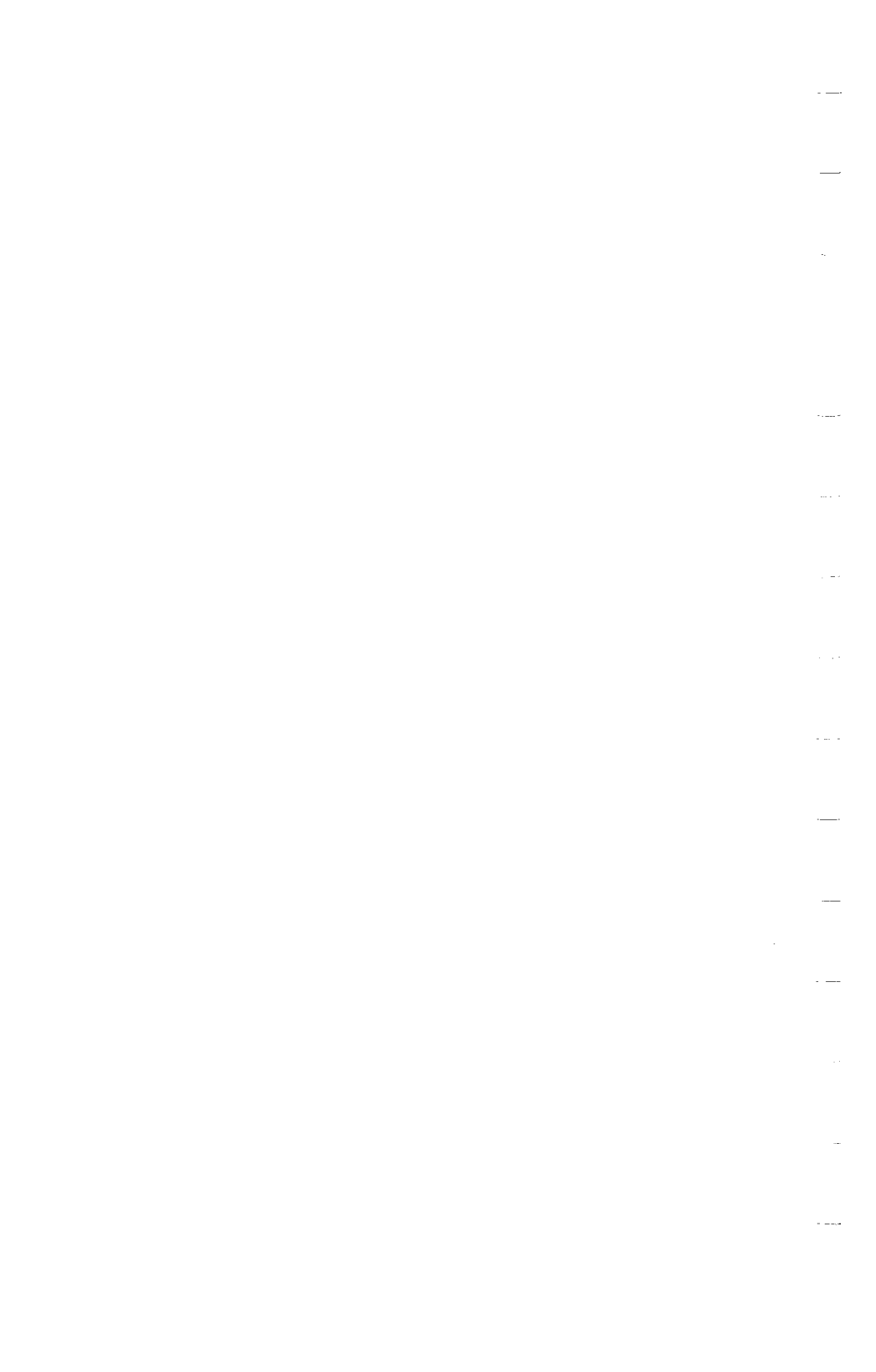
- Removing the disk that contains the batch file being executed will cause an error.
- Immediately upon executing one batch file, you may call another. To do so, put the second file's name (without its extension) as the last command in the first file.

- In a batch file, you may want to refer to a file whose name contains a percent sign. For example, you may want to include a command that copies a file called `Abc%.exe`. As you know, the batch file normally interprets the percent sign as a replaceable parameter. To indicate that this is not the case, you must include a second percent sign. The batch file command, then, might be this:

```
copy a:\user\abc%%.exe b:\user
```


Part 2
MS-DOS Commands





MS-DOS COMMANDS

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 may 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.

Quick Reference To MS-DOS Commands

COMMAND	PURPOSE
APPEND	Sets a data file path
ASSIGN	Reassigns drive names
ATTRIB	Sets or displays file attributes
BACKUP	Copies disk files to floppy disk
BREAK	Alters <input type="checkbox"/> CONTROL <input type="checkbox"/> C operation
CHDIR/CD	Changes current or home directories
CHKDSK	Checks MS-DOS diskettes
CLS	Clears the video screen
COMMAND	Starts a new command processor
COPY	Copy, append or combine files
COPYDOS	Copy system files to application diskettes
CTTY	Switches input/output to device
DATE	Enter or change system date
DEL	Deletes files from specified directory
DIR	Displays files from specified directory
DISKCOMP	Compares two diskettes
DISKCOPY	Makes copies of floppy diskettes
DISKTYPE	Determines format of non-standard diskettes
ECHO	Controls display of lines in batch files
ERASE	Deletes specified files
EXE2BIN	Converts <i>exe</i> files to binary format
EXIT	Exits from commands to previous level
FC	Compares contents of two files
FDISK	Partitions a hard disk
FIND	Searches for specified text

COMMAND	PURPOSE
FOR	Executes several items with one command
FORMAT	Prepares disks for system use
GOTO	Jumps to selected routine in batch files
GRAPHICS	Copies screen graphics to printer
HFORMAT	Prepares a hard disk for system use
HSECT	Formats hard sectors on a hard disk
IF	Allows conditional execution in batch files
JOIN	Joins a disk drive to a pathname
KEYTXX	Replaces the current keyboard program
LABEL	Modifies a volume label
LF	Suppresses the line feed after a carriage return
LPSETUP	Enables a printer filter that allows pagination
MKDIR/MD	Creates a new directory
MLFORMAT	Formats a hard disk partition
MLPART	Creates a hard disk partition
MODE	Sets video, printer, and communication parameters
MORE	Stops screen scroll — awaits key press
PATCH	Modifies a disk file
PATH	Specifies path to external commands
PAUSE	Suspends batch execution, displays message
PRINT	Background printing of up to 10 files
PROMPT	Creates a new system prompt
RECOVER	Recovers bad sectors on a disk

COMMAND	PURPOSE
REM	Allows comments in a batch file
RENAME/REN	Change (Rename) filenames
REPLACE	Updates previous versions of a file
RESTORE	Copies files from diskette to hard disk
RMDIR/RD	Deletes a specified directory
SELECT	Changes the current country code
SET	Establishes a common replacement parameter
SHARE	Installs file sharing and locking
SHIFT	Moves replaceable parameters in batch files
SHIPTRAK	Parks hard disk heads
SORT	Sorts input from keyboard or a file
SPOOLER	Sends commands to the printer spooler
SUBST	Substitutes a virtual drive name for a pathname
SYS	Transfers system files to disk
TIME	Displays or sets system time
TREE	Display all disk directories and files
TYPE	Displays contents of specified file
VER	Displays MS-DOS version number
VERIFY	Verifies that files are intact
VOL	Displays volume label of specified disk
XCOPY	Copies files and directories

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 as part of a pathname.

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

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

- `COPY` is the command name you type to call up the `COPY` function. Although you may enter a command name in either uppercase or lowercase, for your convenience command names are presented in uppercase in this manual.
- *source pathname* is a required parameter. Parameters not surrounded by brackets are essential to the operation and must be included in the command line. In this case *source pathname* is the name of the file you wish to copy. Example: `COPY myfile`.
- [*target pathname*] is an optional parameter. Include descriptions or values for parameters surrounded by brackets only if you wish to change the default description or value. In this case if you omit *target pathname*, MS-DOS will give the new copy the same filename as the original file.
- [/A], [/B], and [/V] are optional *switches*. Again, include them in the command line only if you wish to change the default settings of these switches. In this case 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 may 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 may use the MS-DOS editing and function keys. (See Chapters 1 and 8 for editing information.)
- Commands execute only after you press **ENTER**.
- Pressing **CTRL C** 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 **HOLD**. To resume, press **HOLD** again.

Organization of MS-DOS Commands

The MS-DOS commands in Part 2 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 command 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

BOLD	A command keyword. Unless noted otherwise, you can type an entire command in any combination of uppercase and lowercase letters.
UPPERCASE	Information you type exactly as it appears.
<i>lowercase italics</i>	Variable words, letters, characters, or values.
[]	An optional command parameter. (Do not type the brackets.)
... (ellipsis)	The preceding parameter can be repeated.
KEYNAME	A key on your keyboard.

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

Also, certain commands listed in this manual are valid only for a specific version of MS-DOS or apply only to hard disk systems. These commands are indicated as follows:



MS-DOS
V 2.11
Command



MS-DOS
V 3.2
Command



MS-DOS
hard disk
Command

If you are not sure which version of the operating system your computer uses, type `VER` .

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`.

COMMAND REFERENCE

APPEND

External



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

Sets a search path for data files.

Parameters

pathname specifies the drives and/or directories that MS-DOS searches for a data file.

Notes and Suggestions

- You can specify more than one path to search by separating each path with a semicolon (;).
- If you use the Append command with no options, MS-DOS displays the current data path.

Append ;

sets no data path. MS-DOS searches only the working directory for data files.

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

Code	Function
0FH	Open File (FCB)
23H	Get File Size
3DH	Open Handle

- You can use the Append command across a network to locate remote data files.
- If you are using both the MS-DOS Assign and Append commands, you must type the Append command before Assign.

Example

Suppose you want to access data files in a directory called LETTERS on drive B and in a directory called REPORTS on drive A. To do this, type:

```
append b:letters;a:reports
```

ASSIGN

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

Reassigns drive names (letters). MS-DOS will route requests for a disk drive to the reassigned drive.

Parameters

drive1 is the drive letter you wish to reassign.

drive2 is the drive letter to be given *drive1*.

Notes and Suggestions

- ASSIGN lets you use application programs on drives other than those for which they were specifically designed. A program designed to only operate on Drives A and B could be used on fixed Drive C.
- This command does not require a colon after the drive letter.
- If you enter ASSIGN without parameters, all drives are reassigned to their normal designation (drive letter).
- You should take care when reassigning drives used with MS-DOS or an application program. Some functions will expect to find data or information on a specific drive and ASSIGN could make the data or information inaccessible.
- DO not use ASSIGN with JOIN, PRINT, or SUBST.
- DISKCOMP, DISKCOPY, and LABEL do not allow assigned drives as parameters.

Example

```
assign a=c b=c
```

routes all requests for Drive A or Drive B to Drive C.

ATTRIB

External

3.2

ATTRIB [*read*][*archive*] *pathname*

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

Parameters

read can be either +R or -R. +R sets the read-only attribute of a file to ON. -R sets the read-only attribute of a file to OFF.

archive can be either +A or -A. +A sets the archive attribute of a file to ON. -A sets the archive attribute of a file to OFF.

pathname is the path to the file you wish to reference.

Notes and Suggestions

- If an application opens a file with read *and* write permission, ATTRIB can force a read-only condition to allow file sharing over a network.
- To display the attribute of a specific file, type

```
attrib pathname 
```
- The wildcard characters *.* can be used to print the attributes of all files on a specific drive.
- If you omit the optional parameters, ATTRIB displays the attribute settings of the file specified by *pathname*.

Example

```
attrib +r myfile.txt
```

turns on the read-only attribute of Myfile.txt.

```
attrib +a myfile.txt
```

turns on the archive attribute of Myfile.txt.

BACKUP**External**

BACKUP [*pathname*] *drive*: [/S] [/M] [/A]
 [/D:*date*]

Copies (backs up) one or more files from a hard disk to floppy disks (diskettes). To use files that are on backup diskettes, you must use the RESTORE command to copy them back to hard disk. Do not try to use them otherwise.

Parameters

pathname specifies the file you want to back up.

drive: specifies the disk to receive the files.

/S causes BACKUP to copy all files in the specified directory and all directories below it.

/M causes BACKUP to copy only those files that have been modified since the last backup.

/A tells BACKUP to add the files (to be backed up) to the diskette already in the specified drive, rather than prompting you to insert a new diskette.

If you omit the /A parameter, BACKUP asks you to insert a diskette. Then, before backing up any hard disk files, it erases any existing files on the diskette. When the diskette is filled, BACKUP asks you to insert another. Be sure to label each diskette so you know the proper order of the backups.

/D:*date* copies only those files created on or after the date specified (in the *mm-dd-yy* format).

Notes and Suggestions

- Loss of information stored on hard disk, although not likely, can be disastrous—simply because of the amount of information. Therefore, you should always keep and update floppy diskette copies of all hard disk information.
- If you have several files to back up, you may want to create a batch file consisting of the necessary backup commands. Then you can do all the backups simply by running the batch file. (See information on Batch Files in Chapter 4.)

- BACKUP works best if BUFFERS = 5 (or greater) in the CONFIG.SYS file. (See Appendix B.)

Examples

```
backup *.* a: 
```

backs up all the files in the current directory. You must first use the FORMAT command to prepare several diskettes for information storage. Then insert the first formatted diskette into Drive A and type the command as above. Whenever it fills a diskette, BACKUP prompts you for another.

```
backup inventory.dat a: /a 
```

adds the hard disk file *inventory.dat* to a backup diskette when the file is in the current directory.

```
backup *.bat a: 
```

backs up all batch (*.bat*) files from the current directory to Drive A. This command erases any existing files on the diskette before copying files to it.

```
backup c:\store1\sales.dat a:/a 
```

backs up the file *sales.dat* that is in the STORE1 directory in Drive C to the diskette in Drive A, without erasing the diskette's existing files.

```
backup *.* a:/m 
```

backs up (to the diskette in Drive A) all files in the current directory that have been modified since the last backup.

```
backup c:\ a:/s 
```

backs up (to the diskette in Drive A) all files in all directories on Drive C.

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 a hard disk to a floppy disk.

Parameters

pathname is the path to the hard disk file you want to back up.

drive: the disk to receive the files.

/S produces a backup of subdirectories also.

/M backs up only those files that have been changed since the last backup.

/A adds the specified files to those already on the backup floppy disk. The existing files on the floppy diskette are not erased.

/P saves the file copies in a “packed” format that conserves diskette space. It creates a subdirectory on the floppy diskette if that is the only way to fill the floppy diskette. **Warning:** IBM™ BACKUP/RESTORE compatibility can be lost if this switch is used.

/D backs up only those files that were last modified at or after the specified date.

/T backs up only those files that were last modified at or after the specified time. *hh* can be 01-12. *mm* can be 00-59. *x* can be a for a.m. or p for p.m.

/L creates a backup log entry in the file specified. If not specified, the file is given the name Backup.log, and is placed in the root directory of the diskette. The first line of the file contains the backup date and times. Each subsequent line contains the filenames and the number of the floppy disk on which the files are stored. This information is useful when restoring a particular file from a floppy diskette. By looking at Backup.log file, you know which disk to specify for restore. On subsequent restore operations, because Backup.log already exists, the current entry is appended to the file.

Notes and Suggestions

- Unless otherwise specified, the old files on a backup floppy's root directory are erased before new files are added.
- If more than one diskette is required, BACKUP prompts you to exchange diskettes when the current diskette is full.
- If you try to backup a shared file that you do not have access to, `PATHNAME\FILENAME.EXT Not able to backup at this time` is displayed.
- BACKUP files contain control data, and are not usable.
- BACKUP exits with `ERRORLEVEL` set as:
 - 0 - Normal completion
 - 1 - No files found to backup
 - 2 - Some files not backed up due to file sharing conflicts
 - 3 - Terminated by {Ctrl-Break}
 - 4 - Terminated by error
- If the target is a fixed disk, the backup files will be placed in a subdirectory named `\BACKUP`. If the target is a floppy, the files are placed in the root directory.
- The source disk should not be write-protected.

Examples

```
backup c: a: /s
```

copies the files in the current directory of Drive C: to a diskette in Drive A. This operation erases any existing files on the Drive A diskette.

```
backup c: myfiles a: /a
```

copies the files 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: /1 /d
```

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

BREAK

Internal

BREAK *switch*

Turns the **CTRL C** check on or off or displays the current setting. The BREAK command affects only application programs. It has no affect at the MS-DOS command level or to BASIC programs.

Parameters

switch can be ON or OFF.

ON tells MS-DOS to check for **CTRL C** from the keyboard whenever an application program makes any type of MS-DOS function call.

OFF tells MS-DOS to check for a **CTRL C** 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 C** check. Use this command before running an application program that uses the **CTRL C** function key. Pressing **CTRL C** will affect your program instead of the operating system.

```
break on ENTER
```

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

```
break ENTER
```

displays the current setting of the **CTRL C** check.

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. *Pathname* must be another 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 may 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 3 data files: one sorted by your customers' last names, another by zip code, and another by subscription expiration date.

You can store all three files in a directory called \MAG-MAIL. 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, you may want to put each in a separate directory (see MKDIR). For example, you may 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.

Examples

```
b:   
chdir \user 
```

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

```
chdir .. 
```

puts you in (changes your current directory to) the parent directory of your current directory, if you are in a subdirectory.

```
chdir \ 
```

puts you in the root directory of the current disk.

```
chdir \bin\user 
```

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

```
chdir 
```

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 current 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 current 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 [*pathname1*] [/F] [/V] [>*pathname2*]

Checks the directory of the disk in the current or specified drive for errors. After checking the directory, CHKDSK displays the proper error messages, if any, and then gives a status report. If you wish, you may redirect the output from CHKDSK to a file.

Parameters

pathname1 specifies an entire disk or an individual file to check. If you specify a file, CHKDSK displays information about both the drive and the file.

/F tells CHKDSK to correct any errors it can and update the disk. Do not use this parameter if you include a *pathname*.

/V causes CHKDSK to display messages while it is running and gives detailed information about any errors it finds.

pathname2 specifies the file to which CHKDSK is to redirect its output. Do not use this parameter if you use the /F parameter.

Notes and Suggestions

- CHKDSK does not wait for you to insert a diskette.
- Run CHKDSK occasionally for each disk, to ensure file integrity.
- Here is a sample report produced by CHKDSK. The numbers vary on different disks.

```
362496 bytes total disk space
 23552 bytes in 2 hidden files
  1024 bytes in 2 directories
331776 bytes in 8 user files
  6144 bytes available on disk

114688 bytes total memory
 88800 bytes free
```

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

- The following errors are corrected automatically by the /F parameter. For more information on these errors, see Part 6.
 - Invalid sub-directory entry
 - Cannot CHDIR to *filename*, tree past this point not processed
 - First cluster number is invalid, entry truncated
 - Allocation error, size adjusted
 - Has invalid cluster, file truncated

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 asks if you want to fix any errors. It cannot do so, however, until you specify the /F switch.

```
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. CHKDSK asks if you want to fix any errors. It cannot do so, however, until you specify the /F switch.

CLS (Clear Screen)

Internal

CLS

Clears the screen.

Notes and Suggestions

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

Examples

```
cls [ENTER]
```

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

COMMAND

External



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

Starts the command processor.

Notes and Suggestions

- This command starts a new command processor (the MS-DOS program that contains all 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 tells the command processor where to look for the COMMAND.COM file if it needs to reload the transient part into memory.

device: lets you specify a different device (such as aux) for input and output. See the CTTY command in this chapter for more information.

/P tells COMMAND.COM not to exit to a higher level.

/E:*size* specifies the environment size, where *size* is the size in bytes. The size may range between 128 and 32768 bytes. The default value is 128 bytes.

If *size* is less than 128 bytes, MS-DOS defaults to 128 bytes and gives the message:

Invalid environment size specified

If *size* is greater than 32768 bytes, MS-DOS gives the same message, but defaults to 32768 bytes.

/C *string* tells COMMAND.COM to execute the command or commands specified by *string* and return. /C can only be the last parameter.

COPY

Internal

COPY *source pathname* [*target pathname*]

[/A] [/B] [/V]

Copies one or more files: (1) to the same directory (as the source), giving the new file a different filename or (2) to another directory on any disk, giving the new file the same or a different filename.

Parameters

source pathname is the file to be copied.

target pathname is the name and destination to be given the newly created file.

If you omit the filename from the target pathname, MS-DOS assumes you want to give the new file the same filename as the source file.

/A tells MS-DOS to treat the file as an ASCII file (also called a *text* file) when used with a source file. MS-DOS copies only to the first end-of-file (EOF) character.

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

/B tells MS-DOS to treat the file as a binary file, such as a program file, when used with the source file. Therefore, MS-DOS copies the entire file. When used with the target file, it tells MS-DOS not to add an EOF character to the end of the file.

Each of these switches (/A and /B) affects the file immediately preceding it in the command line and all files following, until another /A or /B is encountered.

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. Notice that this default is the opposite of that for the COPY/APPEND and COPY/COMBINE commands.

/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

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

- *Append* one or more files to the end of an existing file, keeping the target file's filename.
- *Combine* files into a new file that has a unique filename.

(See COPY/APPEND and COPY/COMBINE for more information on these uses of COPY.)

- Use the /B parameter with a target file to remove the EOF character from the end of 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 backup copies of files to protect important information.
 - Transfer external commands to another disk: When you receive your system disk, all external commands are in the root directory. (Some of the Version 3.2 external commands and system utility files are on the Supplemental Programs diskette.) Tailor your directory structure to your own needs by moving some of these commands to their own directory.
- You cannot copy a file onto itself. (You cannot copy it to the same directory, giving it the same name.)

```
copy b:memos b:memos [ENTER]
```

will cause an error message. This command:

```
copy memos [ENTER]
```

will also cause an error message. You are trying to copy the file Memos that is in the current directory to another file Memos, also in the current directory.

- Remember, the target disk must be formatted and the target directory must exist.

Examples

```
copy *.* /a b: 
```

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

```
mkdir \bin   
copy format.com \bin 
```

transfers external commands to another directory by first 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: 
```

copies the file personal.dat from the root directory of Drive A to the current directory of Drive B.

```
copy memos.txt /a b:corr.txt 
```

copies the file Memos.txt from the current directory to the current 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:taxes83.dat /a taxes84.dat 
```

copies the file Taxes83.dat from the current directory in Drive B to the current directory on the current disk, 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 current directory of Drive B. MS-DOS does not add an EOF character to the end of the new file, Prog1.exe.

COPY/APPEND

Internal

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

Adds one or more files to the end of another existing file. The files are added in the order in which you list them. When you append files, the original source files still exist separately. The information from them is copied, not moved.

Parameters

target pathname is the file to receive the appendages named in the source pathnames.

source pathname is the file(s) that will be added or appended to *target pathname*.

/A tells MS-DOS to treat the file as an ASCII file when used with the source file. MS-DOS copies only the information up to the first EOF character.

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

/B tells MS-DOS to treat the file as a binary file, such as a program file when used with the source file. Therefore, MS-DOS copies the entire file. When used with the target file this parameter tells MS-DOS not to add an EOF character to the end of the file.

Each of these switches (/A and /B) affects the file immediately preceding it in the command line and all files following, until another /A or /B is encountered.

For example, if you type this:

```
copy onefile.txt /a + myfile.txt + samfile.txt  
ENTER
```

the /A parameter affects all three files. However, if you type this:

```
copy onefile.txt /a + myfile.txt /b +  
samfile.txt ENTER
```

the /A parameter affects only the target file, Onefile.txt.

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

Notice that this default is the opposite of that for the regular COPY command.

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

Notes and Suggestions

- Use Copy/Append to append separate lists. Suppose you have the following three separate files:

biglist	list1	list2
Bob Anthony	Anne Barnes	Jerry Day
Carol Apple	Ted Erickson	Karen Ellis
Curtis Kelly	Mike McAdam	Jennifer Peters
Susan Leonard	Dave Shultz	Larry Thomas

If you type this command:

```
copy biglist + list1 + list2 
```

the new contents of the file Biglist are:

Bob Anthony
Carol Apple
Curtis Kelly
Susan Leonard
Anne Barnes
Ted Erickson
Mike McAdam
Dave Shultz
Jerry Day
Karen Ellis
Jennifer Peters
Larry Thomas

Examples

```
copy b:read.dat + write.dat + print.dat 
```

appends the files Write.dat and Print.dat that are in the current directory to the file Read.dat that is in the current directory of Drive B. MS-DOS adds an EOF character to the end of Read.dat.

```
copy big.tst+*.lst 
```

appends all files that have the extension .lst, and that are in the current directory, to the file Big.tst, that also is in the current directory. MS-DOS adds an EOF character to the end of Big.tst.

```
copy prog1.exe /b + prog2.exe 
```

appends the file Prog2.exe that is in the current directory to the file Prog1.exe that also is in the current directory. MS-DOS copies all information in the file. MS-DOS does not add an EOF character to the end of Prog1.exe.

```
copy oldfile.dat +
```

leaves the Oldfile.dat file intact, but updates the date and time of the file to the current date and time.

COPY/COMBINE

Internal

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

Combines any number of source files into a target file. The files are added in the order in which you list them. When you combine files, the original source files still exist separately. The information from them is copied, not moved.

Parameters

/A tells MS-DOS to treat the file as an ASCII file when used with the source file. MS-DOS copies only the information up to the first EOF character.

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

/B tells MS-DOS to treat the file as a binary file, such as a program file, when used with the source file. Therefore, MS-DOS copies the entire file. When used with the target file, this parameter tells MS-DOS not to add an EOF character to the end of the file.

Each of these switches (/A and /B) affects the file immediately preceding it in the command line and all files following, until another /A or /B is encountered.

For example, if you type this:

```
copy myfile.txt /a + salfile.txt onefile.txt  
ENTER
```

the /A parameter affects all three files. If, however, you type this:

```
copy myfile.txt /a + salfile.txt /b onefile.txt  
ENTER
```

the /A parameter affects only Myfile.txt.

If you omit the /A and /B switches, MS-DOS uses /A. Notice that this default is the opposite of that for the regular COPY command.

/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

- **Warning:** Never combine files into a target file that has the same name as one of the source files. Instead, append other files to the existing file. For example, suppose your current directory contains the files A.lst, B.lst and All.lst. The command:

```
copy *.lst /b all.lst 
```

combines A.lst and B.lst into the All.lst target file, which replaces (destroys) the All.lst source file. The message, Content of destination lost before copy, is displayed.

- Use Copy/Combine to copy existing files into a new file, for example:

```
oldlist1                oldlist2
Anne Barnes             Jerry Day
Ted Erickson           Karen Ellis
Mike McAdam            Jennifer Peters
Dave Shultz            Larry Thomas
```

which can be combined with this command:

```
copy oldlist1 + oldlist2 newlist 
```

MS-DOS creates the file Newlist in the current directory. The file contains the names shown here:

```
newlist
Anne Barnes
Ted Erickson
Mike McAdam
Dave Shultz
Jerry Day
Karen Ellis
Jennifer Peters
Larry Thomas
```

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

Examples

```
COPY B:memos.txt + B:letters.txt B:corr.txt  
ENTER
```

combines the files Memos.txt and Letters.txt that are in the current directory on Drive B. COPY copies information only up to the first EOF character in each source file. Then it places the information in the file Corr.txt in the same directory and adds an EOF character to the end of that file.

```
COPY A:\stats1.dat + A:\stats2.dat  
B:\allstats.dat ENTER
```

combines the files Stats1.dat and Stats2.dat that are in the root directory of Drive A. COPY copies information only up to the first EOF character in each source file. Then COPY places the information in the file Allstats.dat in the root directory of Drive B. MS-DOS adds an EOF character to the end of Allstats.dat.

```
COPY a.com + b.com + B:c.com oneprog.com  
/B ENTER
```

combines the files A.com and B.com that are in the current directory, with the file C.com that is the current directory of Drive B. COPY copies information only up to the first EOF character in each source file. Then it places the information in the file Oneprog.com in the current directory. It does not add an EOF character to the end of that file.

COPYDOS

External



COPYDOS

Copies operating system files onto an application program diskette.

Suggestions and Notes

- Some application programs do not include an operating system on the program diskette. This means you cannot boot (initialize) your computer with that diskette and must use a system diskette to boot your computer. To avoid needing two diskettes when executing a program, transfer the startup system files to your application diskette with COPYDOS.
- COPYDOS works with either protected diskettes (those you cannot backup) or nonprotected diskettes, and with either single- or dual-drive systems.
- Application program diskettes must have 61,000 free bytes of space to store the system files transferred by COPYDOS.
- Make sure any write-protect tabs are removed from the destination diskette (the diskette to receive the system files) before using COPYDOS.
- If you have a single-drive system, MS-DOS will prompt you to insert the destination diskette in drive B. At this point, remove the system diskette from drive A and insert the destination diskette in its place, then continue with the COPYDOS procedure.

Examples

```
copydos 
```

Formats and copies the necessary system boot files to a diskette. The diskette is first formatted. Be sure you do not use the version of COPYDOS on a diskette containing data you do not wish to erase.

```
copydos p 
```

Copies the necessary system files to a diskette to make it bootable. Using COPYDOS in this manner will change the contents of an application diskette but will not erase any data previously contained on the diskette and will not affect its operation. The **p** parameter **must** be in lowercase.

CTTY (Change I/O Device)

Internal

CTTY *device*

Changes the input/output (I/O) device to the device specified. Normally, input comes from the keyboard and output goes to the screen.

Parameters

device is a device which can be either of the following:

AUX — specifies an auxiliary device, normally the RS-232 serial port.

COM1 or COM2 — specifies RS-232 serial Port 1 or Port 2.

CON — specifies the console (input from the keyboard and output to the screen).

Any other character-oriented device capable of both input and output.

Notes and Suggestions

- Because the printer is not capable of input, PRN is not a valid CTTY device.
- CTTY affects only programs that are MS-DOS function calls. Other programs, such as BASIC, ignore the CTTY command.

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. This date is recorded in the directory for any files you create or change. You can also use this 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 one- or two-digit number in the range 1-12. *dd* (day of month) is a one- or two-digit number from 1-31. *yyyy* (year) is a two-digit number in the range 80-99 (1980 is assumed) or a four-digit number in the range 1980-2099.

Notes and Suggestions

- If the month or day is a number less than ten, it is not necessary to include a leading zero (for example, 09/09/84). When MS-DOS stores and displays the date, it includes a leading zero in a one-digit day and excludes it from a one-digit month (for example, it stores 9/09/1984).
- You can separate the date, month, and year with either slashes, hyphens, or periods.
- 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.
- 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 *auto-exec.bat* file. Therefore, you may want to include a DATE command in that 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 can not 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 `PRINT` or `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 `PRINT` again to turn off the *print output* function. Now use `SHIFT PRINT`, `PRINT`, `CTRL P`, or the `PRINT` command as necessary to make your printout.

- In Version 3.2 MS-DOS, you can change the format in which the date is displayed and entered. The `COUNTRY` command in the `CONFIG.SYS` file lets you change the date format to the European standard, **dd-mm-yy**. Refer to Appendix B for more information.

Examples

```
date ENTER
```

displays the current date and prompts you for the new date. For example, if the date is July 16, 1986, MS-DOS displays:

```
Current date is Mon 7-16-1986
Enter new date:
```

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

```
date 07/14/1986 ENTER
```

enters the current date as July 14, 1986.

DEL (Delete)

Internal

See the ERASE command.

DIR (Directory)

Internal

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

Displays: (1) information about all files or the specified files that are in the current directory or in the directory specified by *pathname* or (2) information about the file specified by *pathname*.

Parameters

pathname indicates a specific file. If you omit the *pathname*, MS-DOS lists the directory entries for all files that are in the current directory. You can use the MS-DOS wild cards in the *pathname*.

/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 SPACEBAR.

/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.
- If you do not include a filename extension, DIR assumes the *.** wildcard extension.
- If the filename does not include an extension, add a period to the end of the filename in the DIR command.
- If the Version 3.2 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 B for more information.

Examples

```
dir b:\user\mailsr*.bat 
```

provides information on how much space B:\USER\mailsr*.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 the file you want is not in the root directory, use the DIR command as needed to check the subdirectories.

```
 dir 
```

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

```
dir b: 
```

displays a list of all files in the current 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 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

drive1: is the drive containing the source diskette for the comparison.

drive2: is the drive containing the target diskette.

/1 compares only the first side of the diskettes.

/8 compares only eight sectors per track.

Notes and Suggestions

- The main purpose of this command is to compare diskettes after a DISKCOPY operation to ensure that the two diskettes contain identical data.
- You cannot compare a floppy diskette with a hard disk.
- The /1 parameter compares only one side of the diskettes, even if they are double sided.
- When complete, DISKCOMP will prompt:

Compare more diskettes (Y/N)?

If you press Y to do another DISKCOMP, the comparison will be done on the drives you originally specified, but it will now be based on the sides and sector configuration of the new diskettes. For example, if the new diskettes are double-sided and the previous diskettes were single-sided, a double-sided comparison will be made. If you press N, the DISKCOMP function will terminate.

- You can escape from the DISKCOMP command by typing **CTRL C** or by pressing **BREAK**. The command will delay acting on the **CTRL C** or **BREAK** request until it is performing a diskette read.
- If you omit the *drive2* parameter, DISKCOMP assumes the current drive.

- If no parameters are specified, DISKCOMP will do a single drive comparison using the current drive.
- During the DISKCOMP process, you will be informed of any disk cylinders or tracks that do not match. DISKCOMP will then continue with the comparison.
- Copies of diskettes made using the COPY command will not usually be identical. Data is not necessarily copied in the same position on the target diskette as it was located on the source diskette. Use the DISKCOPY command for mirror-image copies of diskettes.
- If a read error occurs on a track of one diskette, that track is reported as differing.
- You cannot compare a double-sided diskette with a single sided diskette. Trying to do so will cause an error message.
- Diskettes with a different number of sectors per track cannot be compared.
- *Drive1* and *drive2* cannot be virtual drives, such as those created with the Version 3.2 SUBST command.

Examples

```
diskcomp 
```

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

```
diskcomp a: b: /1 /8 
```

compares the diskette on Drive A with the diskette on Drive B. Only one side is compared and only eight sectors of each track are compared.

DISKCOPY

External

DISKCOPY [*source drive*:] [*target drive*:] [/1]

Makes exact copies of the diskette in the *source drive* on the diskette in the *target drive*.

Parameters

source drive: is the drive containing the diskette you want to copy.

target drive: is the drive containing the diskette to receive the copy.

/1 copies only the first side of a diskette.

Notes and Suggestions

- If the source and target drives are the same, DISKCOPY performs a single-drive copy. It prompts you to insert the diskettes at the appropriate times. Press the space bar to continue.
- If you omit the target *drive* specification, DISKCOPY uses the current drive.

If you omit both drive specifications, DISKCOPY 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.)

- After copying, DISKCOPY prompts:

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

If you press Y, DISKCOPY prompts you to insert the diskettes. Then it performs the next copy, using the drives you specified earlier.

If you press N, DISKCOPY exits the COPY command.

- You can use DISKCOPY to duplicate your MS-DOS diskette.

- We suggest that you make copies of all your diskettes—especially your system diskette and application program diskettes—and store your originals in a safe place.
- If the target diskette is not formatted, or if it is formatted differently from the source diskette, DISKCOPY formats it in the same configuration as the source diskette.
- DISKCOPY does not aid in making fragmented disk files contiguous.
- You cannot DISKCOPY between different drive types. Use the FORMAT and COPY or FORMAT and XCOPY commands for this purpose.

Examples

```
diskcopy 
```

copies all information from a diskette in the current drive to another diskette in the current drive. DISKCOPY 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 prompts you to insert the 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 *drive* is not specified, the default drive is used.

Notes and Suggestions

- Using DISKTYPE with floppy diskettes displays the number of sides, tracks, and sectors-per-track of the diskette in the specified drive. The screen also shows the MS-DOS command line that must be typed to format a new diskette with the same format.
- Using DISKTYPE with a hard disk displays the number of heads, cylinders, and sectors-per-track.

Examples

```
DISKTYPE A:
```

displays the following message:

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

```
DISKTYPE C:
```

displays the following message:

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


ECHO

Internal

ECHO [*switch*] [*message*]

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

Parameters

switch can be ON or OFF. ON tells MS-DOS to display batch file commands. OFF tells MS-DOS not to display batch file commands.

message is text the screen displays when ECHO is encountered.

Notes and Suggestions

- If you do not specify a *switch* or *message*, MS-DOS displays the current setting of ECHO.
- 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 restored to its previous setting after a 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.
- Use ECHO OFF when you do not want to clutter the screen with unnecessary information.
- You can turn off ECHO and insert your own messages to be displayed while a batch file is executing.

Example

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 is displayed, then the screen shows:

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

Formatting tracks
-----
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, the commands are not echoed. 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) one or more files from 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

- 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 prompts: Are you sure (Y/N)? If you type Y (ENTER), MS-DOS erases the files. If you type N (ENTER), 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 preparatory to deleting the directory.
 - Erase a file for which you have no further need.
- You cannot erase hidden files or files marked read-only.

Examples

```
erase *.lst 
```

would be used after you combine the files A.lst and B.lst from the current directory into a new file All.lst on a different directory. It would erase all files with the extension .lst in the current directory. You should make sure that there are no other files you wish to keep that have the .lst extension.

Warning: If you include an extension, such as the .lst used here, ERASE does not prompt you before erasing the files, regardless of whether use a wildcard. It prompts only when you use the full wildcard (*.*)

```
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 *.exe* (executable) file, specified by *source pathname*, to *.com* file format (binary 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

source pathname specifies the file to be converted. If you do not include an extension in the source pathname, the extension defaults to *.exe*.

target pathname specifies a file to receive the converted program file. If you omit the drive, EXE2BIN 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, EXE2BIN 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 prompts for the 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.

When 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 current 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 current directory of Drive A.

EXIT

Internal

EXIT

Returns control from a secondary command processor to the invoking program, if one exists. If you have entered a secondary command processor from an application program, use EXIT to return to the application.

Notes and Suggestions

- While running an application program, you may need to use an MS-DOS command. To do so, you must first invoke a secondary command processor from your program. Enter the command you wish to use. When complete, enter EXIT to return to your application program.


Examples

Invoke a secondary command processor from your program, then type your command.

```
dir b:   
exit 
```

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

FC (File Comparison)**External**

 **FC** [/number] [/B] [/C] [/W] *pathname1*
pathname2 [>*target pathname*]

Compares the contents of two files, *pathname1* and *pathname2*, and sends the output to the screen or to the file specified by *target pathname*.

Parameters

number specifies the number of lines that must match for the file to be considered as matching again after FC finds a difference. It can be from 1 through 9. If you don't specify a number, it defaults to 3. Use this parameter only in source comparisons.

/B forces a binary comparison of both files. The two files are compared byte-by-byte, with no attempt to resynchronize after a mismatch. FC displays the 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 of *pathname1* and *pathname2*, respectively. If one file contains less data than the other, FC displays a message to that effect.

For example, if *pathname1* ends before *pathname2* ends, then FC displays:

```
***Data left in pathname2***
```

/C causes FC to ignore the case of letters. All letters in the files are considered uppercase letters. For example:

```
Much_MORE_data_IS_NOT_FOUND
```

matches

```
much_more_data_is_not_found
```

Use the /C parameter only in source comparisons.

/W compresses *white spaces* (tabs and spaces) for the comparison. Thus, if several whites are together on one line, FC considers them as one white space. FC ignores spaces that are at the beginning and end of lines.

For example:

```
___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
```

(Each underscore represents a white space in these examples.)
Use the /W parameter only in source comparisons.

pathname1 pathname2 can be either *source files* (files that contain source statements of a programming language) or *binary files* (files output by the assembler, the linker, or a Microsoft high-level language compiler).

Notes and Suggestions

- FC uses a large amount of memory as buffer (storage) space to hold the source files. If the source files are larger than available memory, FC compares what can be loaded into the buffer. If no lines match in the portions of the files in the buffer, FC displays only the message:

```
*** Files are different ***
```

and returns to the system prompt. For binary files larger than available memory, FC compares both files completely, overlaying the portion in memory with the next portion from disk. This does not affect the command output.

- The two kinds of comparisons are line-by-line and byte-by-byte.

In a line-by-line comparison, FC marks off blocks of lines and then compares the lines within each block. In a byte-by-byte comparison, FC simply displays the bytes that are different.

- If FC finds many differences (involving many lines), it only reports that the files are different. Then it stops.

- Use FC to determine which of two files is current and what changes have been made since the old version.

Suppose your current directory contains these source files. Each letter in the files represents a program line.

name1.src

A
B
M
D
O
S
W
X
Y
Z
R
U
N
Q
T
V

name2.src

A
B
C
L
S
W
X
Y
Z
P
E
N
Q
V

To compare the files line-by-line, type:

```
fc /1 name1.src name2.src 
```

FC begins by marking off the blocks of lines. Whenever FC encounters a match, it ends a block. Therefore, FC blocks off the lines as follows:

name1.src	name2.src
A	A
B	B
M	C
D	L
O	S
S	
W	W
X	X
Y	Y
Z	Z
R	P
U	E
N	N
Q	Q
T	V
V	

FC ignores matching blocks that are before the first mismatch. It also ignores blocks in which at least the specified number of lines match. In this case, the number is one. For each set of blocks that contains a mismatch, however, FC displays the following information:

- -----NAME1.SRC
- The lines in the Name1.src block that differ from those in the corresponding Name2.src block
- The line that is the same (the last line in the block)
- -----NAME2.SRC
- The lines in the corresponding Name2.src block that differ from those in the Name1.src block

- The line that is the same (the last line in the block)

Therefore, for the files Name1.src and Name2.src, FC displays:

```
-----NAME1.SRC
```

```
M  
D  
D  
S
```

```
-----NAME2.SRC
```

```
C  
L  
S
```

```
-----  
-----NAME1.SRC
```

```
R  
U  
N
```

```
-----NAME2.SRC
```

```
P  
E  
N
```

```
-----  
-----NAME1.SRC
```

```
T  
V
```

```
-----NAME2.SRC
```

```
V
```

You can print the differences on the line printer using the same two source files. To do this, type:

```
fc/1 name1.src name2.src >prn 
```

Examples


```
fc /B test1.src test2.src >test3.src 
```

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 
```

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.

FC (File Comparison)

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

Compares the contents of two files or sets of files.

Parameters

pathname1 and *pathname2* specify the files to compare.

/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**n* sets the internal line buffer to the number of lines specified by *n*. 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
last line that matches in the two files
intermediate, non-matching lines in first file
next line that matches in the two files
***** filename2
last line that matches in the two files
intermediate, non-matching lines in second file
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, use the MORE filter to display the output one screen at a time. For example, type:

```
fc /b file1.src file2.src | more 
```

- You can redirect FC-s output to a file by appending `>target pathname` to 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 
```

- 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 
```

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 
```

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, changes, deletes or displays hard disk partitions. Use this command to initially install a hard disk drive, to change the current partitions, or to check the current partitions. When initializing a hard disk, FDISK is used in conjunction with HSECT and HFORMAT or HSECT and FORMAT, depending on the version of the operating system you are using.

Notes and Suggestions

- When FDISK is activated, 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.

- A hard disk can be partitioned to install more than one operating system on it, or for other specialized applications. Using FDISK you determine where and how much of the total disk to allocate for separate DOS systems.
- Up to four partitions can be made on a hard disk. If you do not intend on using more than one disk operating system, or if you are not sure what systems you may use in the future, specify one partition.
- If you have already installed Drive C, choose menu item 5 to verify the status of Drive D.
- When selecting partition parameters, the maximum available space in cylinders will be displayed and you will be requested to enter the partition (number of cylinders) as well as the starting cylinder number.

- If you need to change a previously partitioned hard disk, use the `BACKUP` command to backup your files in the DOS partition to floppy diskettes. After re-partitioning your hard drive, use `RESTORE` to replace the files on the hard drive.

Example

```
fdisk 
```

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)---->
```

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

```
b: 
fdisk 
```

establishes Drive B as the current drive, loads `FDISK` 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 Nonactive. An active partition is booted 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 the specified string of text in one or more files which are specified by one or more pathnames. Capitalization and punctuation in the string must be the same as in the file.

Parameters

pathname is the file to search. If you omit the pathname option, FIND uses text from the standard input device.

string is the text for which to search. Enclose the string in quotes. For example, to search the Book1.txt and Book2.txt files in the current directory for all occurrences of this string:

```
The whiteness of the whale
```

type:

```
find "The whiteness of the whale" book1.txt
book2.txt 
```

MS-DOS displays the lines in the order in which the files are specified. If the string contains quotations, enclose each quotation in double quotes. For example, to find all occurrences of this string:

```
Aye, "Moby Dick" is a whale of a story.
```

that are in the file Book1.txt in the current directory, type:

```
find "aye, ""moby dick"" is a whale of a
story."book1.txt 
```

/V causes FIND to display all lines that do not contain the string. Do not use this 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 this 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.)
- 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  
ENTER
```

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 names of the files in the home directory of the disk in Drive B that do not contain the string DAT.

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* once for each item in *set*.

Parameters

set is a list of items. Each item in the list is separated with a space or comma. You can include wildcards in the list. Files, directories, and disk drives are only a few of the items you can list.

%C is any one-character variable except 0-9 that MS-DOS uses to represent the items in *set*. DO tells MS-DOS to do the *command*.

%%C is the same as %C, except the second percent sign is required when the command is in a batch file.

command is the command you are executing.

Notes and Suggestions

- Append %C (or %%C for a batch file) to the *command* to execute it sequentially for each item in *set*, substituting that item into the *command*. For example:

```
for %g in (*.dat *.dta) do dir %g
```

This first displays a directory of all .dat files, then displays a directory of all .dta files. However, if you omit the %g at the end of *command*, FOR executes the *command* **one time** for each item in *set*, but does not substitute each item into the *command*.

```
for %g in (*.dat *.dta) do dir
```

This displays a directory of the entire disk twice (once for each item in *set*).

- 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 
```

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

```
copy con 3dir.bat   
rem This file displays directory listings for  
%1, %2, and %3   
rem it is called 3dir.bat   
for %%f in (%1 %2 %3) do dir %%f   
 
```

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 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*] [/V] [/1] [/8] [/B] [/S]

Prepares the blank floppy diskette in the specified drive for use by defining the tracks and sectors and writing system information onto the diskette. You can format either a blank diskette or a diskette already formatted. If the diskette is already formatted and you reformat it, you lose whatever programs or data are on it.

This command applies to floppy diskette only. To prepare a hard disk, you must use the HFORMAT command.

Parameters

drive is either Drive A or Drive B. For example, if you specify Drive A, FORMAT prompts:

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

Insert into Drive A the diskette to be formatted. Press the space bar.

If you omit the drive specification, MS-DOS formats the diskette in the current drive.

/V causes FORMAT to prompt for a volume label after the diskette is formatted. If you use this parameter, you can enter a label of a maximum of 11 characters, or you can press **ENTER** to bypass the prompt. Entering a volume label helps keep track of your diskettes.

/1 performs a single-sided format of a diskette. If you omit /1, FORMAT performs a double-sided format.

/8 causes a format of eight sectors per track. If you omit /8, FORMAT uses nine sectors per track.

/B formats eight sectors per track and allocates space for the hidden system files. Use the /B parameter to create a diskette on which you can place any version of MS-DOS using that version's SYS command. Do not use /B with /S or /V.

/S causes FORMAT to copy the hidden system files and COMMAND.COM to the diskette after it is formatted. This makes the newly formatted diskette a system diskette.

Notes and Suggestions

- Use **FORMAT** to prepare a new disk to receive data. Before you can store information on a new disk, you must prepare it using **FORMAT**. After formatting, record the volume label and date of creation. Store this information in a safe place. It helps you estimate how long a diskette has been in use.

While formatting the diskette, **FORMAT** displays:

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

Note that **FORMAT** prompts you to insert the disk. If the disk is already in the drive, press the space bar. **FORMAT** displays:

```
Formatting...
```

When the format is complete, MS-DOS displays this message:

```
nnnnn bytes total disk space
nnnnnn bytes available on disk

Format another (Y/N)?
```

Type **Y** to format another diskette in the same drive.
Type **N** if you do not want to format another diskette.

- Use **FORMAT** to *clear* an old disk to receive new data. First do a **DIR** command and make sure you don't need any files on the disk. Then use **FORMAT** to erase all old information and lock out any flawed sectors that have developed. **FORMAT** puts the system information back on the disk and leaves the *good* sectors available for information storage.
- The switches **/1**, **/8**, and **/B** are not valid for use with 3½-inch diskettes.

Examples

```
format 
```

formats the disk in the current drive.

```
format a: 
```

formats the disk in Drive A.

```
format b:/s/v 
```

formats the disk in Drive B, makes it a system disk, and asks you for a volume label. You can enter a label that is a maximum of 11 characters (for example, DISKONE), or press if you don't want to label the disk.

FORMAT

External



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

Prepares the disk in the specified drive for use by defining the tracks and sectors and writing system information onto the disk.

Parameters

drive: is the disk drive to format.

/B formats a diskette with eight sectors per track and allocates space for the hidden system files. If you use this switch, you can place any version of MS-DOS on the diskette using that version's SYS command. If you do not use /B, you can place only MS-DOS version 3.2 on the diskette with the SYS command. Do not use /B with /S or /V.

/V prompts for a volume label after the disk is formatted. A volume label can be as many as 11 characters long. A volume label identifies the disk.

/1 performs a single-sided diskette format. (If the drive in which you are formatting is double-sided, and you do not specify this switch, you cannot use the newly formatted diskette in a single-sided drive.) If you omit /1, FORMAT performs a double-sided format.

/8 formats eight sectors per track on a diskette if you do not specify /8, FORMAT defaults to nine sectors per track. (You should note that FORMAT always **creates** nine sectors per track; the /8 switch tells MS-DOS to **use** only eight of those sectors.)

/S copies the hidden system files and the COMMAND file to the disk after it is formatted. The newly formatted disk becomes a system disk. /S must be the last switch in the FORMAT command.

Notes and Suggestions

- This command initializes the directory and file allocation tables on a disk. You must use this command to format all new disks before MS-DOS can use them.
- Only the /V and /S switches are valid for a hard disk format.

- You must specify the *drive* that you want to format.
- FORMAT uses the drive type to determine the default format for a disk.
- When you format a hard disk, FORMAT prompts you with the following:

Enter current Volume Label for drive (x:)

Press if your hard disk does not have a volume label. (If your hard disk has never been formatted, or if it has a bad boot sector, FORMAT will not prompt you for a volume label.)

If the volume label that you enter does not match the label on the hard disk, FORMAT displays the message:

Invalid Volume ID Format failure

Otherwise, FORMAT continues:

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

If you want to format your hard disk, type Y . If you do not want to FORMAT your hard disk, type N .

- If the operating system isn't on the default drive, FORMAT prompts you to insert a system disk in the default drive (or in Drive A if the default drive is a hard disk drive).
- When formatting is complete, FORMAT displays a message showing the total disk space, any space marked as defective, the total space used by the operating system (when you use the /S switch), and the space available for your files.
- Formatting destroys any previously existing data on a disk.
- FORMAT ignores drive assignments created with the ASSIGN command.
- You should not use FORMAT with drives used in the ASSIGN, JOINT or SUBST commands.

- You cannot format drives over the network. FORMAT returns the following errorlevel codes:

- 0 = Successful completion
- 3 = Terminated by user (Control-C)
- 4 = Fatal error
(any error other than 0, 3, or 5)
- 5 = "N" response to hard disk prompt, "Proceed with format (Y/N)?"

You can check these exit codes by using the errorlevel option with the IF batch processing command.

- Use FORMAT to prepare a new disk to receive data. Before you can store information on a new disk, you must prepare it using FORMAT. After formatting, record the volume label and date of creation. Store this information in a safe place. It helps you estimate how long a diskette has been in use.
- While formatting a diskette, FORMAT displays:

```
Formatting...
```

When the format is complete, FORMAT displays this message:

```
Format complete
nnnnn bytes total disk space
nnnnnn bytes available on disk

Format another (Y/N)?
```

Type Y to format another disk in the same drive. Type N if you do not want to format another disk.

Examples

```
format a: 
```

formats the disk in Drive A.

```
format b: /v /s 
```

formats the disk in Drive B, makes it a system disk, and asks you for a volume label. You can enter a label that is a maximum of 11 characters (for example, DISKONE), or press if you don't want to label the 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 8 characters; it ignores the rest. When a batch file executes, the screen does not display its labels. Therefore, you can use labels to include comments in a batch file.

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.

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   
 
```

This batch file 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:

```
Chkdest 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. If it does, 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.

GRAPHICS**External****GRAPHICS** *ptype* [/R] [/B] [/CR] [/LF]

Reproduces a graphics screen on a printer

Parameters*ptype* 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 black to be printed as black, and white to be printed as white. If **/R** is not specified, **GRAPHICS** prints black as white and white as black.

/B causes the background color to be printed when the CGP-220 printer type is selected. If **/B** is not selected, the background is not printed.

/CR causes an end-of-line character to be executed as a carriage return.

/LF causes an end-of-line character to be executed as a line feed.

If neither **/LF** nor **/CR** are selected, both a carriage return and a line feed are executed.

Notes and Suggestions

- The /CR and /LF switches only affect printers that are Tandy DMP standard, or are PC compatible. If you find that a line feed is not performed when required, or that an unwanted blank line is printed between text, use the appropriate /CR or /LF switch to correct the problem.
- Once GRAPHICS is initialized, reset the computer to turn off this function. If GRAPHICS is not initialized, pressing causes a text screen to be printed. When GRAPHICS is initialized, causes the image of a graphics screen to be printed. Pressing a second time terminates a current print operation.
- Graphics images are printed in a maximum of 8 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.
- Once initialized, screen printing can be invoked using assembly instruction INT 5. The GRAPHICS command can be invoked any number of times with different parameters. If the printer type remains the same, only one copy of the program remains in memory.
- If you omit *p*type 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 printer
[C] For PC compatible printer
[D] For PC compatible printer in TANDY MODE
[E] For TANDY DMP-110 printer
```

Type the letter that corresponds to your type of printer.

HFORMAT**External****HFORMAT** [*drive*][*/S*][*/V*][*/B*]

Prepares a hard disk for use and, optionally, makes it an operating system disk by writing the system files to it. The hard disk can be either blank or already formatted when you enter the HFORMAT command. If the disk is already formatted, all information on it is erased when you reformat.

Parameters

drive is a drive specification of C: or greater. If you omit the drive, HFORMAT uses Drive C.

/S causes HFORMAT to copy the hidden system files and COMMAND.COM from the MS-DOS system disk in Drive A to the hard disk after it is formatted. This makes the newly formatted disk a system disk.

/V causes HFORMAT to prompt for a volume label after the disk is formatted. If you use this parameter, you can enter a label of a maximum of 11 characters, or you can press to bypass the prompt.

/B causes HFORMAT to prompt for information it needs to *lock out* flawed areas on the hard disk so that MS-DOS never writes to them. (Having a small number of flawed areas is not uncommon.)

Notes and Suggestions

- Appendix D outlines the entire hard disk setup procedure. Refer to this appendix as necessary when setting up your hard disk.
- Before you can store information on a hard disk, you must prepare it using HFORMAT. During formatting, you have the option of transferring the system files to the hard disk, making it a system disk. System files can only be transferred using HFORMAT */S*. Type COPY A:*. * C: to copy all other files from your system diskette to your hard disk.
- Immediately after you enter the HFORMAT command, HFORMAT displays a message such as:

Press any key to begin formatting drive

Press the space bar. HFORMAT displays:

```
Formatting cylinders
-----
-----
-----
-----
-----
-----
```

When the format is done without error, each dash, which represents an area on the disk, becomes a period. A question mark in place of a period indicates that a portion of the disk contains flawed areas. HFORMAT locks out these areas so that MS-DOS never writes to them.

When the format is complete, MS-DOS displays this message:

```
nnnnnnnn bytes total disk space
nnnnnnnn bytes in bad sectors
nnnnnnnn bytes available on disk
```

Examples

```
hformat /s 
```

formats Drive C and transfers the system files to it. After removing the Drive A diskette and resetting your computer, you can start up your system under hard disk control.

HSECT

External**HSECT** [*drive*:]

Formats hard sectors on a hard disk. Use **FDISK** and **HFORMAT** to complete the hard disk format process.

Parameters

drive is the hard disk to be formatted. Your primary hard disk is Drive C, a secondary hard disk is designated Drive D. If you omit *drive*, HSECT uses Drive C.

Notes and Suggestions

- Appendix D outlines the entire hard disk setup procedure. Refer to this appendix as necessary when setting up your hard disk.
- To begin the hard disk format process, power up your computer and insert an MS-DOS diskette into Drive A. Insert the Hard Disk Utilities diskette into Drive B, and boot your system. (If you only have one floppy drive, replace the MS-DOS diskette with the Hard Disk Utilities diskette after booting.)
- **Warning:** if you want to format Drive D, be sure to specify D:. Otherwise, HSECT uses the default, Drive C, destroying all data on that drive.
- To cancel the HSECT operation, press **[ESC]** or **[CTRL] [C]**.
- At the end of the process, the screen displays:

```
Format completed!
```

Example

```
hsect c:
```

loads HSECT from the current drive, and formats the hard sectors of Drive C. The prompt, `Format completed!`, appears when the process is finished.

HSECT

External



HSECT

Hard formats track and sector information on the hard disk. Use FDISK and FORMAT to complete the hard disk format process.

Notes and Suggestions

- Appendix D outlines the entire hard disk setup procedure. Refer to this appendix as necessary when setting up your hard disk.
- To begin the hard disk format process, power up your computer and insert an MS-DOS diskette into Drive A. Insert the Supplemental Programs diskette into Drive B, and boot your system.
- After you enter the HSECT command, the following prompt appears:

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

To format the primary hard disk drive, type `c` . To format a secondary hard disk drive, type `d` .

- 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. Use HSECT only when you are installing an operating system on your hard disk for the first time.

Type `n` to exit the formatting procedure and return to the Main Menu, or type `y` to continue.

- After you verify 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 
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 listed on the map, press to begin the formatting procedure.

- To cancel the HSECT operation, press or .
- Do not interrupt the program while it is formatting the drive. When the format is complete, HSECT displays `Format completed!`

Examples

```
hsect 
```

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

```
b:hsect 
```

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 the command is to execute only if the program previously executed by COMMAND has an exit code of *number* or higher.

string1 = *string2* indicates that the command is to execute only if *string1* and *string2* are identical after parameter substitution. Strings may not have embedded separators.

EXIST *pathname* indicates that the command is to execute only if the file specified by *pathname* exists.

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 
goto end
: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 filename, 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.lst goto g`
sends program execution to the line after :G if all.lst exists.

JOIN



JOIN [*drive:*] [*pathname*] [/D]

Joins a disk drive to a specific pathname.

Parameters

drive: is the drive to be joined.

pathname is the drive and directory to which *drive* is joined.

/D turns off the effects of a previous JOIN command. You cannot use /D with the *pathname* parameter.

Notes and Suggestions:

- JOIN links the root of a specified drive to a different pathname. If the path already exists, it must be empty.
- If the pathname does not exist, JOIN tries to make a directory with that pathname. 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 only join a drive to a first level subdirectory, such as `join d: c:\memos`. However, `join d: c:\memos\june` is incorrect.
- 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 B: to the directory name A:\HARDWARE. Now, when you type `a:\hardware`, MS-DOS assumes you mean the root directory of Drive B.

KEYTXX

External



KEYTXX [/US]

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

Parameters

/US tells KEYTxx to convert character scan codes to US scan codes.

Notes and Suggestions

- The available keyboard programs are:

KEYTFR	France	KEYTSP	Spain
KEYTGR	Germany	KEYTUK	United Kingdom
KEYTIT	Italy		
- The /US parameter is required to operate some application programs that are configured for a US keyboard.
- To return to the U.S. keyboard layout from a KEYTXX program, press **CTRL** **SHIFT** **F1**. (Use the shift key on the left side of the keyboard.) To select the resident KEYTXX program with true scan codes, press **CTRL** **SHIFT** **F2**. To select the resident KEYTXX program with US scan codes, press **CTRL** **SHIFT** **F3**.
- Although you can have more than one KEYTXX program resident in memory at one time, you can operate only in the last loaded version.
- You can use the **SELECT** command to create a DOS diskette that loads whichever KEYTXX program you specify. Refer to the **SELECT** command for information on how to do this.
- To select the characters marked on the front of the key caps, press **CTRL** **SHIFT** and the appropriate character key.
- MS-DOS lets you produce an accented character. To do so press and release the appropriate *dead key* (an accent key that does not produce a character unless it is used in combination with another key). Then press the appropriate letter key. You can use the accent character by itself by pressing the accent key and then pressing the space bar.

Allowable Key Combinations

France: ä Ä ë ì ö Ö ü Ü ý â ê î ô û
Germany: á é É í ó ú à è ì ò ù
Italy: ä Ä ë ì ö Ö ü Ü ý â ê î ô û
Spain: á é í ó ú É à è ì ò ù ä Ä ë ì ö Ö ü Ü ý â ê î ô û
UK: no dead keys

LABEL

External



LABEL [*drive:*][*label*]

Lets you create, change, or delete a volume label.

Parameters

drive is the letter designation of the disk drive for the label you want to modify.

label is the volume name.

Notes and Suggestions

- *label* can have a maximum of 11 characters. If you do not include a new volume name in the command line, the program displays the following prompt:

Volume label (11 characters, ENTER for none)?

Add a volume label or change the current volume label by typing a new label consisting of 11 characters or less.

Delete the current label by pressing **ENTER** without typing a label name.

Example

```
label a:mydisk ENTER
```

gives the diskette in Drive A the label MYDISK.

LF

External

LF

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

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.
- Line feed codes are suppressed only if they are immediately preceded by a carriage return.
- LF supports the MODE LFOFF, MODE LFON, MODE DMP, and MODE DWP commands. LPDRVR.SYS also supports these MODE commands.

LPSETUP

External

LPSETUP *switch*, *p*type, *port* [, *page*]

Enables a printer filter that allows pagination. The text for the page header and footer are entered into this filter with special Escape Code Sequences.

Parameters

switch is the filter status:

ON causes the printer pagination filter to become resident and to take control.

OFF temporarily disables this filter.

*p*type is the printer type:

TANDY is a Tandy printer that is not in IBM mode.

IBM is a Tandy or other printer that is set in IBM emulation mode. (See your printer manual for more information.)

port is the parallel printer port this pagination printer filter uses. *port* can be 1, 2, or 3.

page is the starting page number, in the range 1-65535, of the printout. The default page number is 1.

Notes and Suggestions

- You can insert the current date, page number, or time anywhere within the page header or footer. LPSETUP will insert the date, page, and time wherever you place a D+=, P+=, or T+= (respectively) in the header or footer text. Note that the page number will always be substituted for P+=, whether automatic page numbering (on the first line of each page) is turned on or off.
- Remember that D+= and T+= will be replaced by ten and eight characters, respectively. Be careful not to exceed the maximum number of characters per line for your printer when typing in header or footer text.

- The first three parameters must **all** be included on the command line when ON is specified. OFF should be specified alone. For example, LPSETUP ON, IBM, 1 and LPSETUP OFF are valid commands, but LPSETUP ON, IBM is not.
- Each time LPSETUP is turned on, any previous escape codes sent to a printer in IBM mode (or to LPDRVR.SYS) that deal with page length or perforation skip are cancelled. Otherwise, there would be conflicts between the pagination attempted by LPSETUP and that by the IBM printer or LPDRVR.SYS. If your printer is **not** IBM-compatible, and has been issued escape codes an IBM printer would not understand, consult your printer manual and cancel those codes to prevent the same printer/LPSETUP pagination conflicts mentioned above. In particular, use whatever code it takes to cancel skip perforation.
- Each time you invoke LPSETUP with the ON parameter, the line number is set to 1, and the page number is either set to the page number value in the command line, or to 1 if you omit that parameter. All other pagination settings (number of lines per page, number of lines to skip over page perforation, etc.) are **not** changed until a new escape code is issued for that particular pagination setting.
- You must re-type the LPSETUP command each time the clock inside your computer rolls over to a new 24-hour period (day) to insure LPSETUP has the current date.
- If the parallel port number you specify is not available (because the appropriate parallel board is not installed in your machine), you will be told so, and the entire command will be aborted!

- The characters `CNTL L` (form feed) and `CNTL Z` (end-of-file marker) are trapped by LPSETUP, and are never actually sent to the printer. A `CNTL L` will cause LPSETUP to automatically print a footer on the current page, skip perforation, and print the header on the next page, in that order. A `CNTL Z` will also cause LPSETUP to print the current page footer and issue perforation skip, but the new page number is set back to 1, and the header on this new page is not printed until the first character to be printed on that page is sent. Hence, issue a `CNTL L` (by the `LPRINT CHR$(&HC)`; command in BASIC, or by using the `PRINT` utility) after each printed file when you want to print your next file at the beginning of a new page, but you do not want the page number of that next page set back to 1; Issue a `CNTL Z` (by the `LPRINT CHR$(&H1A)`; command in BASIC) when you want the same results, but with the page number set back to 1. Note, however, that a `CNTL Z` has no effect if issued **immediately after** a `CNTL L`; at least one line of text must be printed beyond the page header before a `CNTL Z` will be heeded!
- Each time you change the pagination format of LPSETUP using one or more escape code sequences, be sure to set your printer to top of form on a new page, and re-invoke the LPSETUP command.
- The maximum page number is 65535.
- The page header and footer may contain up to 1320 characters, each. If you try to exceed this limit, you will get a warning message.
- You may invoke LPSETUP with or without LPDRV.R.SYS in the CONFIG.SYS bootup file.

Escape Code Sequences recognized by LPSETUP:

ESC 99(Hex) *first char second char ... last char 0*
Input for new header string composed of each separate character.

ESC 9A(Hex) *first char second char ... last char 0*
Input for new footer string composed of each separate character.

ESC 9B(Hex) 0 or 1

Input to toggle automatic page numbering on and off, respectively. Note that embedded page numbering by the "P+ =" delimiter will always print the page number at the designated location.

ESC 9C(Hex), 0 or 1

Input to specify the end-of-line character for each line printed: 0=line feed after each carriage return (ASCII 0D hex); 1=line feed only after each line feed character (ASCII 0A hex). Whether AUTOMATIC page numbering is on or off.

ESC 'C' 1 to 255

Input to select number of lines per page.

ESC 'N' 1 to 255

Input to set number of lines to skip over perforation per page.

ESC 'O'

Input to turn off page perforation skip.

Examples

```
lpsetup on,tandy,1,650
```

makes resident a portion of LPSETUP, and tells the resident portion that the computer is online to a Tandy printer on parallel port number 1, and that you want page numbering to begin at 650.

```
lpsetup off
```

turns off pagination by LPSETUP.

To send escape code sequences in formatting your printed page, refer to this BASIC example.

```
10 LPRINT CHR$( &H1B ); CHR$( &H99 ); 'PRINT
   ESC,99,<HEADER TEXT>,0
20 LET HEAD$="*** THIS IS THE HEADER
   ***"+CHR$( &HD )+CHR$( &HA )+"SECOND LINE"
30 LPRINT HEAD$;CHR$( &HD );CHR$( &HA );
40 LPRINT "ON THIS 3RD LINE OF THE HEADER, DATE:
   D+=; TIME: T+=; PAGE: P+=";
50 LPRINT CHR$( &HD );CHR$( &HA );
60 LPRINT CHR$( 0 );
70 LPRINT CHR$( &H1B ); CHR$( &H9A ); 'PRINT
   ESC,9A,<FOOTER TEXT>,0
80 LET FOOT$="*** THIS IS THE FOOTER
   ***"+CHR$( &HD )+CHR$( &HA )+"LINE 2"
90 LPRINT FOOT$;CHR$( &HD );CHR$( &HA );
100 LPRINT CHR$( 0 );
110 LPRINT CHR$( &H1B );CHR$( &H9B ); 'PRINT ESC,9B,0
   OR 1
120 LPRINT CHR$( 1 ); 'TURN ON AUTOMATIC PAGE
   NUMBERING.
130 LPRINT CHR$( &H1B );CHR$( &H43 ); 'PRINT ESC,
   'C',<# LINES PER PAGE>
140 LPRINT CHR$( 66 ); ' (66 LINES/PAGE
   IN THIS EXAMPLE).
150 LPRINT CHR$( &H1B );CHR$( &H4E ); 'PRINT ESC,
   'N',<# LINES TO SKIP PERF
160 LPRINT CHR$( 2 ); ' (SKIP DN, SKIP
   2 LINES IN THIS EXAMPLE).
170 LPRINT CHR$( &H1B );CHR$( &H9C ); 'PRINT ESC,9C,<0
   OR 1> TO SET END-OF-LINE
180 LPRINT CHR$( 0 ); 'CHARACTER
   (CARRIAGE RET, IN THIS EXAMPLE).
190 PRINT " "
200 PRINT "OK...NOW WHEN YOU RETURN TO DOS, SET
   YOUR PRINTER TO TOP OF FORM, AND ISSUE"
210 PRINT "THE 'LPSETUP' COMMAND AGAIN TO INSURE
   LINE AND PAGE COUNTS ARE RESET!"
220 PRINT "...AFTER THAT, YOU MAY BEGIN PRINTING."
230 PRINT "PRESS <ENTER> WHEN YOU ARE FINISHED
   READING THIS MESSAGE."
240 LINE INPUT " ";A$
250 SYSTEM
```

MKDIR (Make Directory)

Internal

MKDIR *pathname*

MD *pathname*

Makes a new directory. You use the MKDIR command to create a hierarchical directory structure.

Parameters

pathname tells MS-DOS under which directory to create the new directory and specifies the name to give it. The *pathname* can be either relative or absolute.

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 that you can use 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 \USER\JOE in your \USER directory.

```
md letters 
```

creates the subdirectory LETTERS under the current directory. If the current directory is \USER\JOE, MS-DOS creates the directory USER\JOE\LETTERS.

```
md b:letters 
```

creates the subdirectory LETTERS under the current directory of Drive B. If the current directory is \USER\JAN, MS-DOS creates the directory \USER\JAN\LETTERS.

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

- Appendix D outlines the entire hard disk setup procedure. Appendices B and C include information on installing MLPART.SYS. Refer to these appendices as necessary when setting up your hard disk.

Examples

```
mformat 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).

The MLPART command is used in conjunction with the MLFORMAT command and the MLPART.SYS device driver. It is intended for use with hard disks that have a capacity of 35 or more megabytes. The MS-DOS partition must start in the first 32 megabytes of a hard disk, but DOS2 partitions enable you to make use of the remainder of the hard disk's space.

Notes and Suggestions

- Appendix D outlines the entire hard disk setup procedure. Refer to this appendix as necessary when setting up your hard disk.
- 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 video monitor, line printer and communication interface parameters.

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

Shifts the video screen left or right. *characters* is the desired character width (40 or 80). *shift* can be R (right) or L (left). T produces a video test screen for evaluating the shift. A prompt asks if you wish another shift. Answer Y (yes) or N (no).

MODE *linefeed*

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

MODE *printer*

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

MODE *scan*

Sets video scan lines. *scan* can be either 200 or 225.

MODE [*video*][*characters*]

Sets video mode and characters-per-line. *video* can be BW (black and white) or CO (color). *characters* can be 40 or 80.

MODE COLORMAP [*old color new color*]

Changes the video palette color to be displayed in text or graphics modes from *old color* to *new color*. That is anytime a program tries to display something on the screen in *old color*, it is changed to *new color*. If both the *old color* and *new color* parameters are omitted, all colors are reset to their original colors. If *old color* and *new color* are the same, then that color is reset to its original color and all other colors retain their settings. The following colors are available for COLORMAP:

Black	Blue	Green	Cyan
Red	Magenta	Yellow	Gray
Dark Gray	Light Blue	Light Green	Light Cyan
Light Red	Light Magenta	Light Yellow	White

MODE COMnumber: [*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, or 960. The default is 300.
- parity* can be N (no parity), O (odd parity), or E (even parity). The default is E.
- databits* can be either 7 or 8. The default is 7.
- stopbits* can be either 1 or 2 stopbits. The default is 1.
- P tells the printer to continuously retry to output on timeout errors.

To reset a RS232 port to the default values, type: **MODE COM *number*** , where *number* is the RS232 port number. The default values are set and displayed on the screen.

MODE FAST

Sets the CPU speed to 7.16 MHz (the default).

MODE LPT1:*characters* [/*type*][,P]

Sets printer characters-per-line. *characters* can be 80, or 132. *type* is the printer type: DMP for dot matrix printers or PC for PC-compatibles. The default is DMP. /P causes continuous retry on timeout errors.

MODE LPT1:*timeout*

Sets the line printer timeout delay. *timeout* can be LONG (2 minutes) or SHORT (45 seconds).

MODE LPT1: = COMnumber:

Redirects parallel printer output to the specified RS232 channel. *number* can be 1 or 2. Initialize the selected RS232 channel using **MODE COM*number*:** before redirecting printer output.

MODE MONO [*switch*]

Sets the computer to use color-oriented software with a monochrome monitor. Software color requests are translated to black, white, and intense white. If you omit the *switch* MODE MONO asks if you have a 350 or 200 scan line monitor (Choose 350 only if your computer supports an optional video board and you install a Mono/Text video board.)

switch can be:

- ON turns on MODE MONO.
- OFF turns off MODE MONO.
- 200 turns on MODE MONO.
- 350 turns on MODE MONO and changes the active video adapter to Mono/Text.

MODE SLOW

Sets the CPU speed to 4.77 MHz. (The default is 7.16 MHz.)

MODE TV

Sets the computer to use a color TV. (Sets 200 video scan lines, color video mode, and 40 characters per line.)

Examples:

```
mode 40 l t 
```

shifts the video display one character left and produces test characters.

```
mode l f o f f 
```

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

```
mode d w p 
```

sets printer parameters to use a Tandy daisy wheel printer.

```
mode 225 
```

sets 225 video scan lines.

```
mode c o 80 
```

sets the color video mode with 80 characters per line.

```
mode colormap black blue 
```

sets blue as the color to display when an application program requests black.

```
mode com1: 300 n 8 1 
```

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

```
mode fast 
```

sets the CPU speed to 7.16 MHz.

```
mode lpt1 : 132/pc 
```

specifies a line printer column width of 132 for a PC-compatible printer.

```
mode lpt1: long 
```

sets a two-minute line printer timeout delay.

```
mode lpt1:=com1 
```

redirects printer output to serial Port 1.

```
mode mono on 
```

sets the computer to use color-oriented software with a monochrome monitor.

```
mode slow 
```

sets the CPU speed to 4.77 MHz.

```
mode tv 
```

sets 200 video scan lines, color video mode, and 40 characters per line.

MORE

External

MORE

MORE is a filter that reads from standard input and displays one screen of information at a time. It then pauses and displays the message `_MORE_` at the bottom of your screen.

Press `ENTER` to display another screen of information. This process continues until all the input data has been read.

Examples

```
type myfiles.com|more ENTER
```

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

```
type b:acctspay.dat|more ENTER
```

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 `ENTER` 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

pathname is the file you want to change.

address is the starting address of the data to be changed (in hexadecimal).

data1 is the list of the hexadecimal values to be changed.

data2 is a list of the hexadecimal data values to replace *data1*.

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:filex.exe,1A23,4462A3,4360B2 
```

patches the file FILEX.EXE, located on Drive B. The command changes bytes 1A23 through 1A25 to the hexadecimal values 43, 60, and B2 (assuming that they are originally 44, 62, and A3).

PATH

Internal

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

Sets a command path, which tells MS-DOS the directories or drives in which to search for external commands. MS-DOS always searches the current directory before it searches the paths set by PATH. You can also use the PATH command 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 specifies the drives and/or directories in which MS-DOS searches for external commands. 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.

If you specify PATH; (PATH followed by a semi-colon), MS-DOS resets the search path to null (no path set).

Notes and Suggestions

- **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 the root (\) of Drive A (using **DEL**).

Now, suppose you start up MS-DOS and immediately format a disk in Drive B. Because **FORMAT** is an external command, 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 semi-colons. 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 in the path, 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 batch file and lets you exit the batch file or continue.

Parameters

message specifies a message to display. The *message* and **Strike a key when ready . . .** appear on the screen if **echo** is on.

Notes and Suggestions

- When MS-DOS encounters the **PAUSE** command, while running a batch file, it stops and displays the message:

Strike a key when ready . . .

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 (change diskettes or ready the printer, for example).

Whenever this is the case, include a **PAUSE** command where necessary. For example, suppose you want to list two files that are on different diskettes. Create a batch file by typing:

```
copy con typfiles.bat ENTER
rem this file types the files rental.dat and
sales.dat ENTER
rem it is called typfiles.bat ENTER
type b:rental.dat ENTER
pause ENTER
type b:sales.dat ENTER
F6 ENTER
```

Typfiles.bat displays the file Rental.dat, then pauses and displays the message:

Strike a key when ready...

Change disks and press **SPACEBAR** 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 .lst into the file All.lst. If the file All.lst already exists the copy will destroy the original contents of All.lst.

To avoid destroying All.lst, create this batch file:

```
copy con comblst.bat ENTER
REM This file combines *.lst files into
all.lst ENTER
ECHO OFF ENTER
IF EXIST all.lst ECHO all.lst already exists.
Press [CTRL-C] to end, or ENTER
PAUSE ENTER
ECHO ON ENTER
COPY *.lst all.lst ENTER
F6 ENTER
```

If All.lst exists and you execute the batch file, MS-DOS pauses and displays the message:

```
all.lst already exists. Press [CTRL-C] to end
or Strike a key when ready...
```

If All.lst 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* [/C] [/P]. . .] [/T]

Lets you put a maximum of ten files in the print queue, so that MS-DOS prints them automatically while you process other MS-DOS commands. This is called *background printing*. The printer must be connected and ready.

Parameters

pathname is the name of the file you want to print. The *pathname* can include a drive reference but **cannot** include directories. The file must be in the current directory of the drive specified.

/C deletes (cancels) from the print queue the file that immediately precedes and all files that follow /C in the command line.

/P adds to the print queue (prints) the file that immediately precedes and all files that follow /P in the command line.

/T deletes (terminates) from the print queue all files that are there, waiting to be printed.

Regardless of the number of /C and /P parameters your command includes, each switch always affects the file immediately preceding it. For example, if you type this:

```
print budget /p sales rentals /c 
```

MS-DOS adds the files Budget and Sales, but cancels the file Rentals.

If you omit all parameters, PRINT displays the contents of the print queue.

Notes and Suggestions

- Whenever you wish to print and use your computer at the same time, use PRINT. The /C and /P parameters let you revise the print queue whenever doing so is most convenient. For example, you can queue these ten files:

```
Jan           May           Aug
Feb           June          Sept
March         July           Oct
April
```

by typing:

```
print jan feb march april may june july aug
sept oct 
```

Use your computer while the Jan and Feb files are printing. Then, because these are no longer in the queue, there is room to add the Nov and Dec files. If you have changed your mind about printing Aug, update the print queue by typing:

```
print nov /p dec aug /c 
```

The file March is being printed, and the queue contains these files:

```
April         July           Nov
May           Sept           Dec
June          Oct
```

Examples

```
print /t 
```

empties the print queue.

```
print a:\temp1.tst/c a:\temp2.tst a:\temp3.tst  

```

removes the files A:\Temp1.tst, A:\Temp2.tst, and A:\Temp3.tst from the print queue.

```
print temp1.tst /c temp2.tst /p temp3.tst 
```

removes the file Temp1.tst from the queue and adds the files Temp2.tst and Temp3.tst. PRINT can output either of the following messages to the printer. They serve only to remind you that the printout is incomplete.

```
All files Canceled by operator
```

You used the /T parameter to cancel the printing of all files in the print queue.

```
drive:filename.ext Canceled by operator
```

You used the /C parameter to cancel the current file in the print queue.

PRINT

External



PRINT [*pathname* [/D:*device*] [/B:*size*]
[/Q:*value*] [/C] [/P...] [/T]

Prints a file on a line printer while you are processing other MS-DOS commands (background printing).

Parameters

pathname is the name of the file you wish to print. The *pathname* is a complete path, including drive and directories.

/D:*device* specifies the print device. If not specified, the default device is LPT1 (printer number1). If you do not use /D:*device*, the first time you use PRINT, it displays a prompt that asks for the print device.

/B:*size* sets the size in bytes of the internal buffer. Increasing the value of the print buffers using /B speeds up PRINT operations. You only use this switch once, the first time you run PRINT.

/Q:*value* selects the number of files allowed in the print queue. The number of files can be in the range 4 to 32, the default is 10. You only use this switch once, the first time you run PRINT.

/C turns on the cancel mode. This switch cancels the file immediately preceding /C, and all files following /C and removes them from the print queue.

/P turns on print mode. This switch adds the file you type immediately preceding /P, and all files you list following /P, to the print queue.

/T deletes all files in the print queue (those waiting to print).

Notes and Suggestions

- Wildcards (* and ?) can be used in the filename parameter.
- 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.
- The disk containing the files to be printed must remain in the specified drive until all printing is complete.

- The printer cannot be used for any other purpose while PRINT has data to print. Any attempt to use the printer will result in an Out of paper message.

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 [*prompt-text*]

Changes the MS-DOS system prompt to *prompt-text*.

Parameters

prompt-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.

Character	Prompt
t	The current time
d	The current date
p	The current directory
v	The MS-DOS version number
n	The current drive specification
g	The > symbol
l	The < symbol
b	The symbol
—	A carriage return and line feed
q	The = symbol
h	A backspace
e	The escape sequence

If you omit *prompt-text*, MS-DOS sets the current drive specification as the prompt.

Notes and Suggestions

- 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 $q $t$__DATE $q $d 
```

In this case, the new system prompt is similar to the following:

```
TIME = 0:07:04.07  
DATE = Thu 11-15-1984
```

- 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 B.) For example:

```
prompt $e[7m$n:$e[m 
```

sets the prompts in reverse video mode and returns to normal 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 *pathname*
RECOVER *drive*:

Recovers (1) a file that contains bad sectors, or (2) all files on a disk that contains bad sectors in its directory.

Parameters

pathname specifies the file to recover.

drive: specifies the disk to recover.

Notes and Suggestions

- If you perform the RECOVER procedure on an entire disk, it changes all filenames to the format, *filennnn.rec*, where *nnnn* is a four digit number unique to each file.
- 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 disks 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. Reenter the lost information. If the file is a data file, recovering the file may or may not help.
- If you are a beginner, you may prefer reentering all information rather than trying to recover a file.

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.

Parameters

remark is a message that a batch file displays during execution if ECHO is on.

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.

Examples

```
rem This file is called billfile.bat 
```

The REM statement is displayed if ECHO is on.

```
rem pathname1 replaces %1, pathname2 replaces  
%2
```

displays the REM statement as a message if ECHO is on.

```
copy con 3dir.bat   
rem This file displays directory listings for  
%1, %2, and %3   
rem it is called 3dir.bat   
for %%f IN (%1 %2 %3) DO DIR %%f ,  
 
```

The above command creates a batch file that, when executed, first reminds you to replace parameters %1, %2, and %3 when executing the file. The second remark simply reminds you of the file's name.

RENAME

Internal

REN[AME] *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 filename, MS-DOS leaves the corresponding characters as they were. For example, if you type the following:

```
ren newfile old???? [ENTER]
```

MS-DOS replaces the ? wild cards with the characters that occupied the same position in the original name. Newfile is changed to Oldfile.

- If you also 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:

```
RENAME *.lst *.prn [ENTER]
```

MS-DOS changes only the extension of all .lst files that are in the current directory. The new extension is .prn. The file Suefile.lst, for example, becomes Suefile.prn.

Examples

```
ren b:\user\gl1.dat gl2.dat [ENTER]
```

changes the name of the Gl1.dat file that is in B:\USER to gl2.dat.

```
ren b:mufile ?y???? [ENTER]
```

changes the name of Mufile, a file in the current directory in Drive B, to Myfile.

REPLACE

External

REPLACE *source pathname* [*target pathname*]
[/A] [/D] [/P] [/R] [/S] [/W]

Updates previous versions of files.

Parameters

/A adds new files to the *target* directory instead of replacing existing ones. Do not use this switch with */D* or */S*.

/D replaces files in the *target* directory only if the *source* files are newer than the corresponding *target* files. Do not use this switch with */A*.

/P prompts you before replacing a target file or adding a source file.

```
Replace (filename) ? (Y/N) _
```

/R replaces read-only files as well as unprotected files. If you do not specify this switch, any attempt to replace a read-only file causes an error and stops the replace process.

/S searches all subdirectories of the target directory while replacing matching files. This switch is incompatible with the */A* switch. **REPLACE** never searches subdirectories in the source path.

/W waits for the user to press a key before replacing files. If you do not specify this switch, **REPLACE** begins replacing or adding files immediately.

Notes and Suggestions

- The **REPLACE** command lets you easily update files on your hard disk with new versions of software.
- The **REPLACE** command replaces files in the *target* directory with files from the *source* directory that have the same name. You may use wildcards in source filenames.
- When you specify the */A* switch, **REPLACE** adds to the *target* directory all files that exist in the *source* directory but **not** in the *target* directory.

- As files are replaced or added, REPLACE displays the file-names on the screen; when the replace operation is complete, the screen shows:

```
   NNN file(s) added/replaced
       or
   No files added/replaced
```

- You cannot use the REPLACE command to update hidden files or system files.
- If REPLACE encounters an error, it returns one of the following errorlevel codes:

1	Command line error
2	File Not Found
3	Path Not Found
5	Access Denied
8	Insufficient Memory
15	Invalid Drive
Other	Standard MS-DOS error

You can test for these codes by using the errorlevel option with the batch processing If command.

Examples

Suppose your hard disk, Drive C, contains several files of client names and phone numbers. To replace these files with the latest version of this file that exists on the disk in drive A, you would type:

```
replace a:\phones.cli c:\ /s
```

This command replaces every file on Drive C that is named Phones.cli (in different directories) with the file Phones.cli from the root directory on Drive A.

Suppose you want to add some new printer device drivers to a directory called C:\MSTOOLS, which already contains several printer driver files for a word processor. To do this, you would type:

```
replace a:*.prd c:\mstools /a
```

This command adds any files from the default directory of drive A with an extension of PRD (that do not currently exist in the \MSTOOLS directory on drive C) to C:\MSTOOLS.

RESTORE

External



RESTORE *source drive:* [*target drive:*]
pathname[/S][/P]

Restores one or more files from diskettes to a hard disk. RESTORE copies only files that were stored on diskettes using the BACKUP command.

Parameters

source drive: specifies the drive that contains the backup diskette.

pathname specifies the hard directories and/or files you want to restore.

target drive: specifies the hard disk drive to which you want to restore.

/S restores all files in the directory specified by *pathname* as well as its subdirectories.

/P prompts before restoring read only files and files that have been changed since the last backup.

Notes and Suggestions

- RESTORE works best if BUFFERS=5 (or greater) in the CONFIG.SYS file (See Appendix B).
- As long as you have a complete library of backup diskettes, restoring your hard disk should be straightforward. Suppose you want to restore all the files that were backed up from all directories on Drive C. Insert the first backup diskette into Drive A, then type:

```
restore A: C:\ /S 
```

- If you wish to restore several files but not the entire disk, as in the previous example, you may want to create a batch file consisting of the necessary backup commands. Then you can do all the backups by simply running the batch file.

Examples

```
restore a: c:*.dat/p 
```

restores all files from Drive A that have the extension .dat and that were backed up from the current directory of drive C.

```
restore a: c:\user\store1.dat 
```

restores from Drive A the file Store1.dat that was backed up from the USER directory of Drive C.

RESTORE**External**

RESTORE *source drive:* [*target drive:*]
pathname [/S] [/P] [/B:*date*] [/A:*date*]
[/E:*time*] [/L:*time*] [/M] [/N]

Restores one or more files previously stored with the BACKUP command.

Parameters

source drive: is the hard disk drive or floppy diskette containing the previously backed up files.

target drive: is the hard disk drive or floppy diskette on which you wish to restore the files.

pathname is the directory or file you wish to restore.

/S restores all files in the directory specified by *pathname* as well as its subdirectories.

/P matches the files specification of any hidden or read-only files. Prompts for permission to restore them.

/B restores only those files that were last modified on or before the given *date*.

/A restores only those files that were last modified on or after the given *date*.

/E restores only those files that were last modified at or earlier than the given *time*.

/L restores only those files that were last modified at or later than the given *time*.

/M restores only those files that have been modified since the last backup.

/N restores only those files that no longer exist on the *target drive*.

Examples

```
restore a: c:\ /s
```

restores all files backed up from all directories on Drive C.

```
restore a: c:*.dat/p
```

restores all files from Drive A that have the extension .dat, previously backed up from the current directory of Drive C.

RMDIR (Remove Directory)

Internal

RMDIR *pathname*

RD *pathname*

Removes specified subdirectories from a disk.

Parameters

pathname is the name of the subdirectory to be removed.

Suggestions and Notes

- Whenever you delete or copy all of a directory's files, you can remove the directory to save disk space.
- You can only delete an empty directory (one that does not contain any subdirectories or files).
- You cannot remove the root directory.

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 current directory of Drive B.

```
chdir\user\ann\accts`  
copy drugstor.dat b:\user\sue [ENTER]  
erase drugstor.dat [ENTER]  
chdir.. [ENTER]  
rmdir\user\ann\accts [ENTER]
```

These commands remove the current directory \USER\ANN\ACCTS, which contains the file Drugstor.dat, from Drive B. If you wish to keep the file, Drugstor.dat, copy it to another directory and then use ERASE to delete it from the Drive B current directory. As MS-DOS doesn't allow you to remove the current directory, you use CHDIR to change directories. Then, remove the \USER\ANN\ACCTS directory.

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. If you only specify *country* in the SELECT command, the current configuration is changed to the new country code. If you also specify *keyboard* with or without /US, a new MS-DOS diskette is created. The /US switch can only be used with a *keyboard* parameter. The following table shows the valid country codes:

Country	Country Code	Keyboard Code
Belgium	032	FR
France	033	FR
Germany	049	GR
Italy	039	IT
Portugal	351	SP
Spain	034	SP
Netherlands	031	UK
United Kingdom	044	UK
Australia	061	US
French Canada	002	US
Israel	972	US
Middle East	785	US
United States	001	US
Denmark	045	*
Finland	358	*
Norway	047	*
Sweden	046	*
Switzerland	041	*

*Indicates keyboard driver can be obtained separately.

keyboard specifies a two-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 KEYTxx.COM commands. Be sure that your MS-DOS system diskette is in Drive A and that it contains the KEYTxx.COM file that matches your selection when using this parameter.

/US specifies US scan codes. You may need to set this parameter to operate some application programs that are configured for a US keyboard.

Notes and Suggestions

- When SELECT is used to make an international backup MS-DOS 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. The versions of these files on the original diskette are not copied.

Examples

```
select 044 uk
```

selects the United Kingdom date, time, currency symbol and decimal separator and the US 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** [*name* = [*string*]]

Sets an environmental variable equal to a string.

Parameters

name is the environment variable to assign to. *string*. *name* cannot be numeric.

string is the string you are naming in the environment. If you omit the *string* parameter, SET removes the variable *name* from the environment. *string* cannot be numeric.

If you omit all parameters, SET displays the current environment variables and their settings.

Notes and Suggestions

- This command is useful only in programs you have written.
- MS-DOS reserves a part of memory for environment variables. If you use a *name* that already exists in the environment, SET replaces the old value of *name* with the new *string*.
- You use SET to its greatest advantage if you have several batch files that use the same string. For example, suppose you have several batch files that contain *%filename%*. One might contain the command:

```
type %filename% 
```

Another might contain the command:

```
copy a: %filename% b: 
```

Note: Within the batch file commands, the string is enclosed by percent signs. Within the SET command, it is not.

The first time you run all the batch files, you may want to substitute Myfile for filename. To do this, use the SET command before you begin. Type:

```
set filename=myfile 
```

MS-DOS automatically substitutes each occurrence of filename—in all batch files—with Myfile. Now suppose you want to substitute Samfile for filename. Reset filename by typing:

```
set filename=samfile 
```

To void the setting, type:

```
set filename= 
```

- The SET command lets you write versatile batch files. An example is to use the SET and the PROMPT commands together to set up a series of easy-to-change system prompts. Suppose you have four system prompts, all of which you use often. Rather than repeatedly entering the PROMPT command with parameters, use batch files to do the work for you.

First, assign each prompt a unique parameter that is to represent it in a batch file.

- promptd = default prompt (current drive)
- prompt1 = time and date
- prompt2 = current directory
- prompt3 = version number and a greater-than symbol

Then set a *name* for each *string* (the codes required by the PROMPT command). For example:

```
set promptd=$n$g 
```

```
set prompt1=The time is $t$__The date is $d  

```

```
set prompt2=$p 
```

```
set prompt3=$v$g 
```

Now create four batch files, each containing a PROMPT command, as follows:

```
copy con promptd.bat [ENTER]
rem This file changes the system prompt to
the default (current directory) [ENTER]
prompt %promptd% [ENTER]
[F6] [ENTER]
```

```
copy con prompt1.bat [ENTER]
rem This file changes the system prompt to
the current time and date [ENTER]
prompt %prompt1% [ENTER]
[F6] [ENTER]
```

```
copy con prompt2.bat [ENTER]
rem This file changes the system prompt to
the current directory [ENTER]
prompt %prompt2% [ENTER]
[F6] [ENTER]
```

```
copy con prompt3.bat [ENTER]
rem This file changes the system prompt to
the version number and greater-than symbol
[ENTER]
prompt %prompt3% [ENTER]
[F6] [ENTER]
```

Now you can quickly change your prompt to any of these four simply by executing the appropriate batch file. For example, execute `promptd.bat` by typing:

```
promptd [ENTER]
```

The file replaces the `promptd` name with the `ng` string and changes the system prompt accordingly.

If you want to use prompts other than those set, you can either set them or change an existing setting. Remember, the settings continue to affect these files and other batch files until you change them or reset or turn off the computer.

- **Note to Experienced Users:** You can use SET to affect application programs, as well as batch files. Refer to the documentation on “Program Segment Prefixes” in your computer’s *Programmer’s Reference Manual* for more information.

Examples

```
set drive=b: 
```

assigns the environment variable *name*, drive, to the B: *string*.

```
set drive 
```

voids the current setting for the variable *name* drive.

```
set tty=vt2 
```

sets your TTY (console) value to VT52 until you use another SET command to change it.

```
set 
```

displays the current environment variables and their settings.

SHARE

External



SHARE [/F:*space*] [/L:*locks*]

Installs file sharing and locking for active networking.

Parameters

/F:space allocates file space (in bytes) for recording 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 is 2048 bytes.

/L:locks specifies the number of locks allowed. The default is 20.

Notes and Suggestions

- Once you use SHARE in an MS-DOS session, SHARE checks all read and write requests.
- MS-DOS uses SHARE only when networking is active. Include the SHARE command in the AUTOEXEC.BAT file if you wish to install shared files at startup.

Example

```
share /f:4096 /l:30
```

doubles the default number of bytes for the storage of file sharing information, and increases the number of allowable locks to 30.

SHIFT

Internal

SHIFT

Lets you use more than the usual ten replaceable parameters (0%-9%) in batch file processing.

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, you must set an environment variable equal to %0 before you use the SHIFT command.

- 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. 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. This way you can use SHIFT to gain access to more than ten 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.

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.
- After running SHIPTRAK, immediately turn off the system. Wait at least 15 seconds before turning 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 
```

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 standard input device, sorts the data, and writes it to the standard output device.

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.

Janet	King	2	8th St.	Lincoln	NE	68502
Samuel	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 /+2 <mail.dat>sortmail.dat 
```

SORT sorts the contents of Mail.dat, starting at Column 2, and outputs the sorted data into the file Sortmail.dat.

Examples

```
sort /r <unsort.txt >sort.txt 
```

reads the file Unsort.txt, sorts the information in reverse alphabetical order, and 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, and 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 driver exists in your CONFIG.SYS file.

Parameters

printer 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.
- Refer to Appendices B and C for information on installing the SPOOLER.SYS device driver.

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 *pathname*.

Parameters

drive: is the virtual drive name.

pathname is the pathname you want to refer to as *drive:*.

/D deletes the association between *drive:* and *pathname*.

MS-DOS displays the names of the current virtual drives if you omit all parameters.

Notes and Suggestions:

- The virtual *drive:* must be in the range A: to LASTDRIVE:. If the LASTDRIVE command is not in your CONFIG.SYS file, SUBST defaults to E: as the virtual *drive:* name. (Refer to Appendix B for more information on LASTDRIVE.)
- The virtual *drive:* cannot be the same as the current drive.
- The virtual *drive:* cannot be the same as a drive reference included in *pathname*.
- SUBST creates or deletes a *virtual drive* by associating a *pathname* with a drive letter.
- When MS-DOS sees a drive that was created with SUBST, it replaces the reference with the new *pathname*.

Examples:

```
subst d: b:\usr\fred\forms
```

creates virtual drive D for the *pathname* B:\USER\FRED\forms. Now, instead of typing the full *pathname*, type D: to go to this directory.

SYS (System)

External

SYS drive:

Transfers two hidden system files from the current disk to a formatted diskette in the specified drive.

Parameters

drive: is the drive containing the diskette to receive the files.

Notes and Suggestions

- The two hidden system files must be the first two files on the target diskette. If there is no room on the diskette or in the diskette's directory, the SYS program does not work.
- If the target diskette is blank, SYS converts it to a system diskette. If the target diskette contains old system files, SYS updates the system.
- COMMAND.COM (the command processor) is not transferred. You must use the COPY command to transfer COMMAND.COM.

Examples

```
sys b: 
```

transfers the system files from the current diskette to the diskette in Drive B.

TIME

Internal

TIME [*time*]

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 may want to include a TIME command in that file.)

Parameters

time specifies the time to set in the *hh:mm:ss.cc* format.

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 fields that follow default to zero.

If you omit the entire time parameter, 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

- You can omit any part of the *time* parameter. However, if you include a part of the *time* parameter, you must include all previous parts. For example, if you include the seconds (*ss*), you must also include hours and minutes.
- 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 can not 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 of how long it takes to run a particular program.
- You can change the time format by adding the `COUNTRY` command to your `CONFIG.SYS` file. Refer to Appendix B 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.

```
time 0:0:0.5 
```

sets the time to one-half (50/100) second past midnight.

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 displays all files in all levels of subdirectories.

Examples

```
tree b: 
```

displays all the directories at all levels contained on the diskette in Drive B.

```
tree a:/f>prn 
```

lists to the printer all subdirectories and filenames on the diskette in Drive A.

```
tree a:/f>adir.fil 
```

writes all the subdirectories and files on Drive A to be placed in the disk file Adir.fil on the current directory of Drive A.

TYPE

Internal

TYPE *pathname*

Displays the contents of the specified file. TYPE makes only one change to the file's format. It expands tabs to spaces consistent with a tab stop at every eighth column.

Parameters

pathname is the name of the file to list.

Notes and Suggestions

- A display of binary files sends control characters (such as bells, form feeds, and escape sequences) to the standard output device.
- Use TYPE to see if you need to change a file. If so, you can use EDLIN to make the change. (See Part 3 for information on EDLIN).

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 current 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.

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 the Radio Shack Customer Service Department.

Example

```
ver 
```

displays the version number.

VERIFY

Internal

VERIFY [*switch*]

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 the data written to disk can be read without error.

Parameters

switch can be ON or OFF.

ON/OFF remain in effect until a program executes a SET VERIFY system call or until you use the VERIFY command again. When VERIFY is on, it verifies all writes to disk.

If you omit all parameters, MS-DOS returns the current setting of VERIFY.

Notes and Suggestions

- By keeping VERIFY on, you always know that data written to disk can be read back. Note, however, that this slows operation.

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. Unlike DISKCOPY, XCOPY does not require that the source and target disks have the same format. You can XCOPY between different disk drive types or different media types.

Parameters

source pathname is the name of the drive, directories, and/or file you want to copy. If you omit the directory part of *source pathname*, XCOPY copies the file from the current directory. If you omit the filename part of *source pathname*, XCOPY uses the wildcard *.* , and copies all files in the directory.

target pathname is the name of the drive, directories, or file to which you are copying. If you omit the directory part of *target pathname*, XCOPY copies to the current directory. The default filename is *.* .

/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:*date* copies only those source files modified on or after *date*. The date format varies, depending on the country code that you are using.

/E copies all empty directories of the specified directory. 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 to confirm that you want to copy each source 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` verifies each target file as it is written to be sure it is identical to the source file.

`/W` waits to begin 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 XCOPY.
```

Notes and Suggestions

- `XCOPY` 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 does not already exist, and copies the file `Payroll` to it.

- If you are using Version 3.2 MS-DOS, 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]
```

This 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.

Examples

```
xcopy a: b: /s /e [ENTER]
```

copies all the files, directories, and subdirectories (including empty subdirectories) on Drive A to Drive B.

```
xcopy jobstat b:\status\ /v 
```

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 
```

copies all read-only files in the current directory of Drive A to Drive B.

```
xcopy a: b:\modfiles\ /d:11/15/86 
```

copies from the current directory of Drive A all files modified on or after November 15, 1986, and places them in the \MOD-FILES directory on Drive B.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

THE LINE EDITOR (EDLIN)

Although the line editor EDLIN and the MS-DOS Commands editing function are separate utilities, many of their operation keys are identical. This chapter provides a quick reference for these keys. Chapter 8 provides sample sessions and instructions for command-line editing. Understanding the command editing features will greatly assist you in learning the full editing features of EDLIN, described in Chapter 9.

Function Keys

You can accomplish many editing functions by using special keys. Following is a chart that illustrates each key and its function.

Function	Key(s)	Description
Copy <i>char</i>	[→]	copies a character from the template to the command line and displays it.
Delete <i>char</i>	[DELETE]	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.
Copy all	[F3]	copies all remaining characters and displays the entire command line.
Insert	[INSERT]	enters the insert mode. Press [F3] to exit.

Replace template	<code>F5</code>	makes the new line the new template, but does not execute the command (accepts the line for more editing).
Void line	<code>ESC</code>	voids the current input. Leaves the template unchanged. Type and enter a new line, or press <code>ENTER</code> to display the system prompt.
Enter line	<code>ENTER</code>	makes the new line the new template and sends it to the requesting program.
End-of-file	<code>F6</code> or <code>CTRL Z</code>	puts an end-of-file character in the template.

EDITING COMMANDS

MS-DOS has several special editing features to help you change commands. Learning to use these features can save you time.

The Template

You edit commands in MS-DOS through a special storage area, 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. Only one command at a time is stored—the last command entered. You can then recall the command from the template for editing or reexecution.

Display and Change

Assume you have two files, Prog.com and Prog.asm, in the root directory of your MS-DOS diskette. 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 **[INSERT]**, press the space bar, then press **[F3]**. The result is:

```
TYPE prog.asm__
```

To execute the new command, press **[ENTER]**.

Saving in the Template

Suppose you misspelled TYPE as BYTE, but had 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:

DELETE **DELETE** **→** **INSERT** **Y** **P** **F3**

Pressing **DELETE** twice deletes the first two characters (*B* and *Y*). The **→** key copies the third letter *T*. **INSERT** enters the insert mode so that you can insert the letters *Y* and *P*. **F3** copies the rest of the template.

DELETE 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.

EDITING FILES

The techniques for editing command lines also apply to editing files. 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 of a maximum of 254 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

You can create your own file with EDLIN by typing:

```
edlin pathname 
```

where *pathname* refers to either an existing file (that you wish to edit) or one 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, you would give a path-name to the desired directory. This command, for example, tells EDLIN to create the file in the current 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 1 to tell EDLIN to display Line 1. The screen looks like this:

```
*1
1:*This is a sample file
1:*__
```

You can enter no more than 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 typing 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 it the filename and extension (if any) you specified when you created it.

Editing an Existing File

Your newly created file, or any EDLIN file, can now be recalled. 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 section). 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 the RENAME command 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 samfile 
```

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* (→)

Using the Copy *Char* function, →, copy characters from the template to the new line.

1. Press → once. The screen looks like this:

```
1:*This is a sample file
1:*T__
```

2. Press → three times. The screen looks like this:

```
1:*This is a sample file
1:*This__
```

Each time you press →, EDLIN copies another character from the template to the new line.

3. To continue to the next example, type 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. Type 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. Type **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 (**DELETE**)

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 **DELETE** four 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. Type **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 first *a* have been 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. Type (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. Type **CTRL C** to continue to the next example.

Example 7: Insert (**INSERT**)

The **INSERT** 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 *f*. The screen looks like this:

```
1:*This is a sample file
1:*This is a sample __
```

3. Now press **INSERT** to turn on the insert mode. Type *edit* followed by a single space. Then press **INSERT** 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. Type **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. Type **[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 type *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.

EDLIN COMMAND REFERENCE

In addition to the editing keys, EDLIN also supports many powerful commands that let you manipulate an entire file or several lines at a time.

Command Format And Rules

Most EDLIN commands follow a certain format and conform to certain rules. Following is a brief review:

- Except for End, Edit, and Quit, EDLIN commands are preceded and/or followed by parameters. The function of parameters is discussed later.
- You may enter EDLIN commands in uppercase or lowercase, or a combination of 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` lists the ten lines immediately preceding the current line, the current line, and the ten lines immediately following the current line.
- You can enter multiple commands in the same line without 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 displays Lines 10-20.

In the second case, type [CTRL][Z] (instead of [ENTER]) after the string to be replaced. For instance, the multiple command:

```
SThis string [CTRL][Z] -5, + 5L [ENTER]
```

searches for the string, *This string*, and then displays the 5 lines that immediately precede the matched string and the 5 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*.
- Control characters (such as [CTRL][C] and [CTRL][Z]) can be inserted into text and even used for the Search and Replace commands. To do this, first type [CTRL][V] and then the desired control character.
- Delimiters (spaces or commas) are required only between adjacent line numbers. You may want to 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 [HOLD]. This halts the display so you can read it. Press [SPACEBAR] again to continue display output.
- You may stop EDLIN commands by typing [CTRL][C].

Command Summary

The EDLIN commands are summarized in the following table.

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 Parameters

Most EDLIN commands require you to specify a line number, a string, or a range of lines to be affected by the command. These are the *command parameters*. Command parameters may vary according to the EDLIN command, but, generally, they are as follows.

line (line number)

This parameter denotes that you must indicate a line number (not always a numeral). Separate line numbers from commands, parameters, and other line numbers with a comma or a space.

You may specify a line in three ways:

- A number less than 65534. If you specify a number larger than the number of existing lines, EDLIN adds a line to the file.
- A period (.), which indicates the current line. This is the last line edited, not necessarily the last line displayed. The current line is always marked on your screen by an asterisk (*) between the line number and the first character.
- A pound sign (#), indicates the last line of a program. For example, the command #1 will begin an insert at the end of your program.

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).

Append Lines (A)

[*number*]A

Adds the specified number of lines from disk to the file being edited in memory. The lines are added at the end of the file.

This command is meaningful only if the file being edited 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*, placing them 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.

The file is renumbered automatically after the copy, and the first line copied becomes the current line. For example:

```
3,9,12C 
```

copies Lines 3-9 to Line 12. 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 to be edited. 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 this entire block of text, at the EDLIN prompt type:

```
1,5,6C 
```

Almost instantly the prompt reappears to indicate that the command was executed. 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 before 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 may 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 may 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 may 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 within 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:

*,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:

*line 1*D or
line1,D

If you do, EDLIN deletes only *line1*.

If you omit *line1* and *line2*, EDLIN deletes only the current line.

D

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 that is required
5: in most cases
.
.
.
26: is to specify
27: a number of lines
28: to be deleted;
29: deleting can often help
30:*to clean up your files.
```

Type:

```
5,25D 
```

The result is:

```
1: This sample file
2: shows how
3: the Delete command functions.
4: All that is required
5:*is to specify
6: a number of lines
7: to be deleted;
8: deleting can often help
9: to 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 to specify
5: a number of lines
6: to be deleted;
7: deleting can often help
8: to 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 can often help
5: to 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:*to clean up your files.
```

Edit Line (line) .

[*line*]

Lets you load the specified line for editing.

EDLIN displays the specified line and — on the next line — 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 omit 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.
- Finally, 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 `[INSERT]`, type the word `number`, followed by a single space, and press `[INSERT]` 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 Copy to transfer the file to another drive.)

If you edited an existing file (rather than a newly created one), EDLIN also saves the original file (unedited file). The original file is given 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 of *.\$\$\$*.

Insert (I)

[*line*]I

Inserts lines of text immediately before the specified line. Also, when you create a new file, you must enter the I command to begin writing text.

Text begins with Line 1. Successive line numbers appear automatically each time you press **ENTER**.

EDLIN remains in the Insert mode until you type **CTRL C**. The line immediately following the insertion becomes the current line and EDLIN automatically increments all line numbers that follow the insertion as necessary.

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 to be edited:

```
1: This is a sample file
2: to show what happens
3: when using the Insert command
4: in your files
```

To insert text before a specific line that is not 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 a new line number. Type:

(how they are renumbered) **ENTER**

Then return to the EDLIN prompt by typing **CTRL C**. Type **L** **ENTER** display the result:

- 1: This is a sample file
- 2: to show what happens
- 3: to the line numbers
- 4: (how they are renumbered)
- 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, type **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 they are renumbered)
- 5:*every time you insert new lines
- 6: when using the Insert command
- 7: in your files

To add lines to the end of the file, type:

8 **I** **ENTER**

The screen shows:

8:*__

Enter the following new lines:

- 8: The Insert command
- 9: can be used
- 10: to add new lines
- 11: to the end of your file

Type `CTRL C` when the number for Line 12 is displayed. 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 they are renumbered)
5: every time you insert new lines
6: when using the Insert command
7: in your files
8: The Insert command
9: can be used
10: to add new lines
11: to the end of your file.
```

List (L)

[*line1*][,*line2*]L

Displays all lines within the range *line1* and *line2*.

Notes

- You can omit *line1*:

*line2*L

If you do, the display begins before the current line and ends with *line2*. The comma is required to indicate that you omitted *line1*.

- You can omit *line2*:

*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.
- If some of your lines are longer than the screen width (require more than 1 display line) they will still be counted as a single line by EDLIN. In other words, if the List command requires more text than the screen can display, the excess will scroll off the top.

Examples

Assume that the following file exists and is ready to be edited:

```
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 the command in this format: *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 the command in this format: *., 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 within 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 renumbers 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 may omit the *line1* parameter, as in ,*line2*P .
- If you do, EDLIN uses the current line number plus 1, unless the current line is Line 1. The comma is required to indicate that you omitted 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 plus 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 may have made to the file.

EDLIN prompts you to make sure you don't 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   
Abort edit (Y/N)___
```

Replace String (R)

[*line1*][,*line2*][?]R[*string1*] [CTRL] [Z] [*string2*]

Replace all occurrences of *string1* with *string2*. The replacement is limited to the lines between *line1* and *line2*.

Notice that you must terminate *string1* by typing [CTRL] [Z], then begin *string2* immediately following the [CTRL] [Z] character. Terminate *string2* by typing another [CTRL] [Z] or [ENTER].

Each time Replace finds *string1*, it replaces the *string1* with *string2*. It displays the line in which it has made a replacement. 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.

? tells Replace to prompt you with O.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*, 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 CTRL Z or ENTER
```

The result is:

```
2: shows how the Replace commor works or how
4: certain words or phrases
6: other words or phrases
```

Note that 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 CTRL Z or ENTER
```

Now, whenever Replace finds the *and* string, you have the opportunity to accept or reject the change.

Note that in the first Replace Command line a space was typed after *and* and *or*. In the second command with ?R, spaces were not typed.

Search Text (S)

[*line1*][,*line2*][?]S**[*string*]**

Searches all lines within 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 finds another match or searches all lines.

- You can omit the *line1* parameter, by typing the command in this format: *,line2Sstring* **ENTER**.

If you do, the search begins at the line that immediately follows the current line.

- You can omit the *line2* parameter, by typing the command in this format: *line1Sstring* **ENTER** or *line1, Sstring* **ENTER**
- If you omit the string, Search uses the last search string that was used 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 may then follow the string with another command, in the next character position.

Examples

Assume that the following file exists and is ready to be edited. 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: may include the optional parameter ? and
9: 2 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 Sand[ENTER]
```

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 [ENTER]
```

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: may include the optional parameter ? and
O.K.? Y
```

The search continues until you press Y 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*, the file contents are inserted ahead of the current line.
- The file to be transferred is read from the current directory of the specified drive or the default drive. If you issued a path when EDLIN was started, that path serves as the current directory. All subsequent Transfer Lines commands use that directory.

Example

```
10TB:myfile 
```

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, files are loaded until memory is 75% full. To edit the rest of your file, you must first write edited lines in memory to disk. Then you can load the unedited lines from disk into memory, using the Append command.

Note

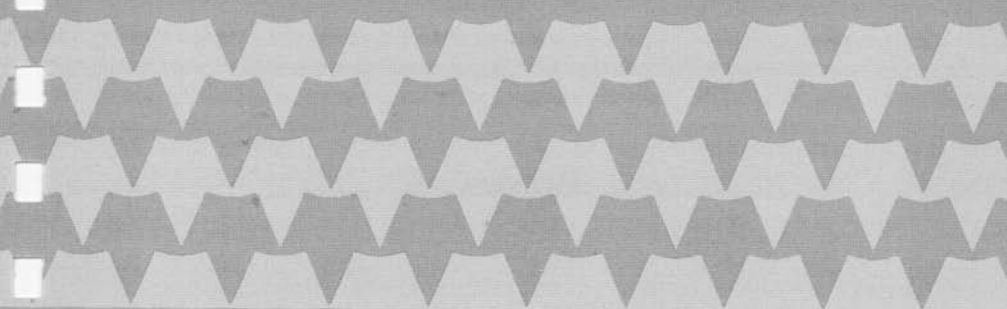
If you omit the number, EDLIN writes lines until 25% of memory is freed. If at least 25% already is freed, EDLIN takes no action. All lines remaining in memory (not written to disk) are renumbered starting with Line 1.

Example

```
100W 
```

writes Lines 1 through 100 to disk and renumbers the file in memory, starting with Line 101 as Line 1.

Part 4
MS[®]-LINK



0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

LINKING OBJECT MODULES

MS®-LINK is an MS-DOS utility program designed to produce ready-to-run machine language code. You should read all of this chapter before you use the linker. Chapter 12 contains technical information for experienced users. LINK error messages are in Part 6.

If you are not going to compile and link programs, you do not need to read this section.

Basic Information

Programs are written in source code. Passing the source code through a compiler or an assembler produces 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 module through the link process. Following is a summary of the linker requirements and functions.

System Requirements

The linker requires the following:

- At least 50K bytes of memory. 40K bytes are for code and data; the remaining 10K bytes are for run space.
- One disk drive, if output is sent to the same diskette from which input was taken.
- Two disk drives, if output is sent to a different diskette.

Because the linker does not allow time to swap diskettes during operation on a one-drive configuration, the use of two disk drives is more practical.

Linking Object Modules and Producing a Run File

The linker lets you link (combine) several separately produced object modules and run them as one program. It produces one relocatable load module, or *run file* (also called an *.exe* or executable file).

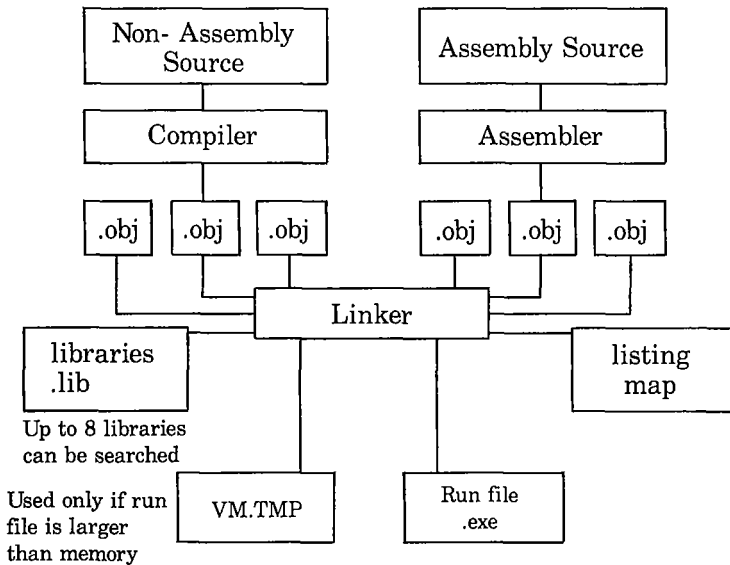
Resolving External References

In addition to containing internal references, modules to be linked may contain “external references” (references to symbols that are defined in the other modules). As it combines modules, the linker resolves all external references (makes sure they are defined). If any external reference is not defined in the object modules, the linker searches up to eight library files for the definition.

Producing a List File

The linker also produces a list file, which shows external references resolved and displays any error messages.

The following diagram illustrates the various parts of the linker’s operation:



The VM.TMP (Temporary) File

If the files to be linked create an output file that exceeds available memory, the linker creates a temporary file, names it *VM.TMP*, and puts it on the disk in the current drive. It displays:

```
VM.TMP has been created.  
Do not change diskette in drive x:
```

where *x* is the current drive.

Once this message is displayed, you should not remove the disk from the current drive until the link session ends. If you do remove the disk, the operation of the linker is unpredictable and the following error message can occur:

```
Unexpected end of file on VM.TMP
```

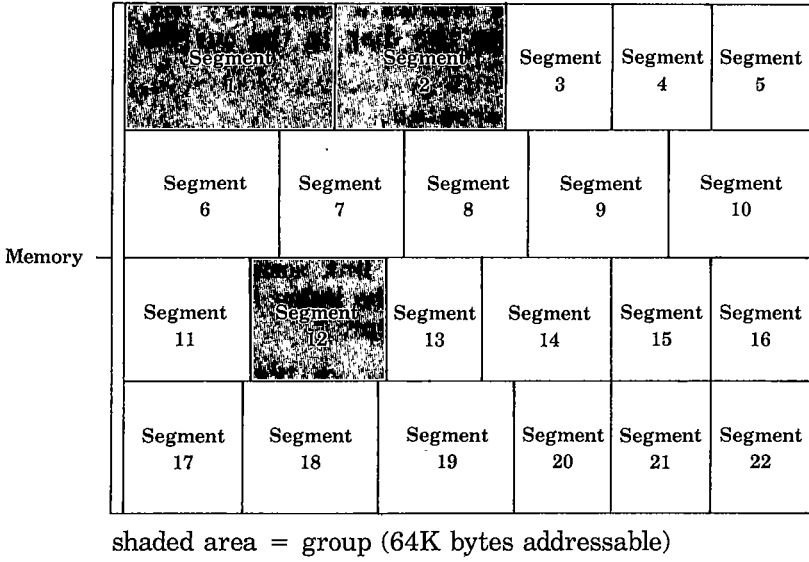
The contents of *VM.TMP* are written to the file you name at the Run File: prompt (see "Command Prompts" in this chapter). *VM.TMP* is a working file only; the linker deletes it automatically at the end of the linking session.

Do not use *VM.TMP* as a filename. If you have a file named *VM.TMP* on the current drive, the linker will overwrite (destroy it) should it create its own *VM.TMP* file.

Definitions

The following explanations of the terms used in this section will 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 must understand the linker and the following definitions. (Chapter 12 also contains useful information on how the linker works.)

In MS-DOS, memory can be divided into *segments*, *classes*, and *groups*. The following diagram illustrates these concepts.



Example

	Segment Name	Class Name
Segment 1	PROG.1	CODE
Segment 2	PROG.2	CODE
Segment 12	PROG.3	DATA

Note that segments must have different segment names but may or may not have the same class name. Segments 1, 2, and 12 form a group; the group address is the lowest address of Segment 1 (the lowest address in memory).

Segment

A segment is a contiguous area of memory up to 64K bytes long. It can be located anywhere in memory on a paragraph (16-byte) boundary. The contents of a segment are addressed by a segment address and an offset within that segment. Segments can overlap.

Group

A group is a collection of segments that fit within a 64K byte area of memory. The segments do not need to be contiguous. The group address is the lowest address of the lowest segment in that group. At link time, the linker analyzes the groups and then references the segments by the group address. A program can consist of one or more groups.

When writing in assembly language, you can assign the names of groups in your program. In high-level languages, the compiler automatically assigns the names. The linker checks to see that the object modules of a group meet the 64K-byte constraint.

Class

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.

All segments assigned to the same class are loaded into memory contiguously. The segments are ordered within a class, in the order they are encountered by the linker in the object files. One class precedes another in memory only if a segment for the first class precedes all segments for the second class in the input to the linker. Classes can be any size, and groups can span classes.

Refer to Chapter 12 for information on how to assign group and class names and on how the linker combines and arranges segments in memory.

Command Prompts

When the linker is started (see “Starting the Linker” later in this chapter), a series of four prompts appears on your display. When you type a response to a prompt and press **[ENTER]**, the next prompt appears.

After the last prompt, the linker begins linking. If it finishes successfully, the linker exits, and the screen displays the MS-DOS system prompt. If an error occurs, the linker displays the appropriate error message.

The linker prompts for the names of the object, run, and list files and for libraries. Use a standard MS-DOS pathname to reference each file.

The prompts are discussed below in the order they appear. Defaults are shown in square brackets ([]) following the prompt. To select a default, press in response to a prompt. The `Object Modules:` prompt is the only one that requires you to type a pathname.

Object Modules [.OBJ]:

- Enter a list (one or more) of the object modules to be linked. If an object module has an extension other than `.obj`, specify it. If you do not, the linker assumes it is `.obj`.
- Separate module names from one another with a blank space or a plus sign (+).
- Remember, the linker loads segments into classes in the order encountered. You can use this information to set the order in which the linker reads the object modules. Refer to Chapter 12 for more information on this process.

Run File [first object pathname.EXE]:

- Enter a pathname. The linker creates a file of that name to store the run file that results from the link session. It assigns the run file the extension `.exe`, even if you specify an extension other than `.exe`.
- If you do not enter a pathname in response to this prompt, the linker uses the first pathname typed in response to the `Object Modules:` prompt. For example, if your first object module to be linked is `Prog1.exe`, the screen shows:

```
Run File [prog1.exe]:
```

To name the run file `B:Payroll`, type:

```
B:payroll/P 
```

This response directs the linker to create the run file `Payroll.exe` on Drive B. Because the `/P` switch is included, the linker pauses to let you insert a new disk to receive the run file.

List File [NUL.MAP]:

- Enter a pathname. If you do not specify an extension, the linker assigns the extension *.map*. The list file contains an entry for each segment in the object modules. Each entry shows the offset for that segment in the run file.
- If you do not enter a response, the linker uses the default value (NUL.MAP) and thus does not produce a list file.

Libraries [.LIB]:

- Enter a maximum of eight library filenames or press . (Only pressing tells the linker to search for default libraries in the object modules.) Library files must have been created by a library utility. Where no extension is given, the linker assumes an extension of *.lib*.
- Separate library pathnames with blank spaces or plus signs (+).
- The linker searches library files in the order listed when resolving external references. When it finds the module that defines the external symbol, the linker processes that module as another object module.
- If the linker cannot find a library file on the disks in the disk drives, it displays:

```
Cannot find library filename
Enter new drive letter:
```

Press the letter for the drive designation (for example, B).

The linker provides three command characters.

- Plus Sign (+). Use the plus sign to separate entries and extend lines as needed. (A blank space can be used instead to separate entries.)

If the response to the Object Modules: or Libraries: prompt is too long to fit on a line, type at the end of the line. The prompt appears again, and you can continue typing the response. After listing all the modules, press .

- Semicolon (;). At any time after the first prompt (Object Modules:), you can select default responses to the remaining prompts. To do so, type ; [ENTER]. This eliminates the need to press [ENTER] repeatedly.

Once you enter the semicolon, you can no longer respond to any of the remaining prompts for that link session. Therefore, do not use the semicolon to skip only a few prompts. To skip prompts, use [ENTER].

- [CTRL] [C]. Use [CTRL] [C] to terminate the link session at any time. If you enter an erroneous response—such as the wrong pathname or an incorrectly spelled pathname—type [CTRL] [C] to exit the linker. Then you can restart the linker.

If you made an error but have not pressed [ENTER], use [BACKSPACE] to delete characters in that line.

You use the seven linker switches to control various linker functions. Switches can be grouped at the end of any response or distributed at the ends of several. To specify a switch, type a slash (/) followed by the switch name (that you can abbreviate to the first letter).

- /DSALLOCATE — The /D switch tells the linker to load all data at the high end of the data segment. If you omit /D, all data is loaded at the low end.

At runtime, the DS pointer is set to the lowest possible address to allow the entire DS segment to be used. Using /D and omitting /H allows any available memory, below the area specifically allocated within DGroup, to be allocated dynamically by the user program. It thus remains addressable by the same DS pointer. This dynamic allocation is needed for Pascal and FORTRAN programs.

Note: Your application program can dynamically allocate up to 64K bytes (or the actual amount of memory available) less the amount allocated within DGroup.

- `/HIGH` — The `/H` switch causes the linker to place the run file as high as possible in memory. If you omit `/H`, the linker places the run file as low as possible in memory.

Important: Do not use `/H` with Pascal or FORTRAN programs.

- `/LINENUMBERS` — The `/L` switch tells the linker to include in the list file the line numbers and addresses from the source statements in the input modules. If you omit `/L`, the linker does not include the line numbers. (If the object modules do not contain line number information, the linker cannot include line numbers.)
- `/MAP` — The `/M` switch directs the linker to list all public (global) symbols defined in the input modules. If you omit `/M`, the linker lists only errors (including undefined globals).

The symbols are listed alphabetically at the end of the list file. For each symbol, the linker lists its value and its segment:offset location in the run file.

- `/PAUSE` — The `/P` switch causes the linker to pause in the link session. This lets you swap disks before the linker outputs the run file. If you omit `/P`, the linker does the link without stopping.

When the linker encounters the `/P` switch, it displays the message:

```
About to generate .EXE file
Change disks <hit ENTER>
```

Press the space bar to resume processing.

Warning: Do not remove the disk that is to receive the list file or the disk used for the `VM.TMP` file, if one has been created.

- `/STACK:size` — The `/S` switch lets you specify the stack size. If you omit `/S`, the linker calculates the required stack size from information in the object modules provided by the compiler or assembler.

size can be any positive value (in decimal) up to 65535 bytes. If you enter a value in the range 1-511, the linker uses 512.

At least one object (input) module must contain a stack allocation statement. If it does not, the linker displays the error message:

Warning: No STACK Segment

- /NO — the /N switch tells the linker not to search the default (product) libraries in the object modules. For example, if you are linking object modules in Pascal and specify /N, the linker does not search the library named Pascal.lib to resolve external references.

You can start the linker by:

- entering responses to the individual prompts as they are displayed.
- including all responses on the command line.
- creating a file to automatically respond to the prompts.

Method 1: Keyboard Responses

To load the linker into memory and display the four prompts, one at a time, type:

```
LINK 
```

These prompts and possible responses are described under “Command Prompts.”

Method 2: Responses on Command Line

Type all prompt responses on the LINK command line. Separate the responses with commas. Use the following format:

```
LINK objlist, runfile, listfile, liblist [/switch...] 
```

objlist is a list of object modules. Use a blank space or a plus sign to separate the module names.

runfile is the name of the file to receive the executable output.

listfile is the name of the file to receive the listing.

liblist is a list of library modules to be searched. Use a blank space or plus sign to separate the module names.

/switch refers to optional switches. Switches can follow any response list. (They can immediately precede any comma or immediately follow *liblist*.)

To select the default for a field, type a second comma with no spaces between the two commas. For example:

```
LINK fun + text + table + care/Pcombine/M,,  
funlist, coblib.lib 
```

loads the linker. Then the object modules Fun.obj, Text.obj, Table.obj, and Care.obj are loaded. The linker then pauses because of the /P switch. When you press the space bar, the linker links the object modules and produces a global symbol map (because of the /M switch). It then creates a run file with the default name Fun.exe, creates a list file named Funlist.map, and searches the library file Coblib.lib.

Method 3: Response File

Type:

```
LINK @filespec 
```

filespec is the name of an automatic response file that contains answers to the linker prompts. An extension is optional. There is no default extension.

This method permits the command that starts the linker to be entered from the keyboard or from within a batch file without requiring you to take any further action.

Before using this option, create an automatic response file. This file should contain several lines of text, each of which is the response to a linker prompt. The responses must be in the same order as the linker prompts discussed earlier. If desired, a long response to the Object Modules: or Libraries: prompt can be typed on several lines. Use a plus sign (+) at the end of a line to continue the response on the next line.

Use switches and command characters in the response file the same way as they are used for responses typed from the keyboard.

When the link session begins, each prompt is displayed, in order, with the responses from the file. If the file does not contain a response for a prompt, the linker displays the prompt and waits for a valid response.

Example

```
fun text table care
/P /M
funlist
cobl.lib
```

The first line in this file tells the linker to load four object modules: Fun, Text, Table, and Care.

The second line omits a name for the run file, thus the linker will use the name Fun.exe. When the linker encounters the /P switch, it pauses to let you swap disks. After doing so, press the space bar to continue. Because the line includes the /M switch, the linker produces a public symbol map (see "Switches").

The third line tells the linker to name the list file Funlist.map.

The fourth line tells it to search the library file Coblib.lib.

Sample Link Session

This sample shows the kind of information that is displayed during a link session.

In response to the MS-DOS prompt, type:

```
link 
```


The system displays the following messages and prompts, with your responses:

```
Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Object Modules [.OBJ]: fun text table care 
Run File [FUN.EXE]: /m 
List File [NUL.MAP]: prn/l 
Libraries [.LIB]: 
```

Notes

- Because you specify /M, the linker displays an alphabetical and a chronological listing of public symbols.
- By responding PRN to the List File: prompt, you redirect your output to the printer.
- Because you specify the /L switch, the linker lists all line numbers for all modules. (The /L switch can generate a large amount of output.)
- Because you press in response to the Libraries: prompt, the linker performs an automatic library search.

Once the linker locates all libraries, the linker map displays a list of segments in the order of their appearance within the load module. The list might look like this:

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

The information in the start and stop columns shows the 20-bit hex address of each segment relative to location zero. Location zero is the beginning of the load module.

The addresses displayed are not the absolute addresses where these segments are loaded. See Chapter 12 for information on how to determine where relative zero is actually located and how to determine the absolute address of a segment.

Because you used the /M switch, the linker displays the public symbols by name and value. For example:

```
ADDRESS      PUBLICS_BY_NAME
009F:0012    BUFFERS
009F:0005    CURRENT_DOS_LOCATION
009F:0011    DEFAULT_DRIVE
009F:000B    DEVICE_LIST
009F:0013    FILES
009F:0009    FINAL_DOS_LOCATION
009F:000F    MEMORY_SIZE
009F:0000    FUN
```

```
ADDRESS      PUBLICS_BY_VALUE
009F:0000    FUN
009F:0005    CURRENT_DOS_LOCATION
009F:0009    FINAL_DOS_LOCATION
009F:000B    DEVICE_LIST
009F:000F    MEMORY_SIZE
009F:0011    DEFAULT_DRIVE
009F:0012    BUFFERS
009F:0013    FILES
```

LINK TECHNICAL REFERENCE

You can link files totaling one megabyte. The output file from the linker (a run file) is not bound to specific memory addresses and, therefore, can be loaded and executed at any convenient address. The relocation information is a list of long addresses that must change when the executable image is relocated in memory. See “Long References” later in this chapter for an explanation.

Definitions

The following terms describe some of the functions of the linker. For definitions of segment, group, and class, see the “Basic Information” section in Chapter 11.

- **Alignment** refers to byte, word or paragraph segment boundaries. You specify the alignment in an assembly language program.

The *byte alignment* tells the linker it can start a segment on any byte boundary (one segment can immediately follow another). *Word alignment* tells the linker to start segments only on even addresses. *Paragraph alignment* tells the linker to start segments only on 16-byte boundaries.

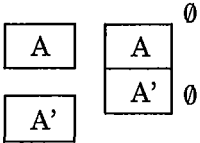
- **Combine Type** is an attribute of a segment. It tells the linker how to combine segments that have the same name or it relays other information about the properties of a segment. Combine types are: stack, public, private, and common (see “How the Linker Combines and Arranges Segments”).
- **Canonical Frame** is the starting address of the first segment in a group of segments. Offsets are calculated from this address.

How the Linker Combines and Arranges Segments

The linker works with four combine types, which are declared in the source module for the assembler or compiler. The types are: private, public, stack, and common. The memory combine type (available in Microsoft's Macro Assembler) is synonymous with the public combine type. The linker does not automatically place memory combine type as the highest segment (as defined in the Intel standard).

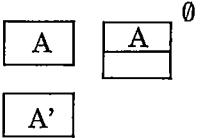
The linker arranges these combine types as follows:

Private



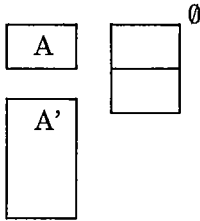
Private segments are loaded separately and remain separate. They may be physically (but not logically) contiguous, even if the segments have the same name. Each private segment has its own canonical frame.

Public and Stack



Public and stack segments of the same name and class name are loaded contiguously. Offset is from the beginning of the first segment loaded through the last segment loaded. There is only one canonical frame for all public segments of the same name and class name. Stack and memory combine types are treated the same as public. However, the stack pointer is set to the last address of the first stack segment.

Common



Common segments of the same name and class name are loaded overlapping one another. There is only one canonical frame for all common segments of the same name. The length of the common area is the length of the longest segment.

Placing segments in a group in the assembler provides offset addressing of items from a single canonical frame for all segments in that group.

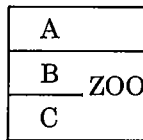
DS:DGROUP

XXXX0H

0 — relative offset

Any number of other segments can intervene between segments of a group.

Thus, the offset of ZOO can be greater than the size of segments in the group combined, but no larger than 64K.



An operand of DGROUP ZOO in assembly language returns the offset of ZOO from the beginning of the first segment (Segment A here).

Segments are partitioned by declared class names. The linker loads the segments belonging to the first class name encountered, then loads all the segments of the next class name encountered, and so on until all classes are loaded.

If your program contains:

A	SEGMENT	'ZOO'
B	SEGMENT	'BAZ'
C	SEGMENT	'BAZ'
D	SEGMENT	'NEW'
E	SEGMENT	'ZOO'

the Linker loads the segments as:

```
'ZOO'  
  A  
  E  
'BAZ'  
  B  
  C  
'NEW'  
  D
```

If you are writing assembly language programs, you can control the order of classes in memory by writing a dummy module and listing it first after the linker's `Object Modules:` prompt. The dummy module declares segments into classes in the order you want the classes loaded.

Warning: Do not use this method with BASIC, COBOL, FORTRAN, or Pascal programs. Allow the compiler and the linker to perform their tasks in the normal way.

Example

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

Make sure you declare all classes to be used in your program in this module. If you do not, you lose absolute control over the ordering of classes.

If you want memory Combine Type to be loaded as the last segment of your program, you can use this method. Add MEMORY between SEGMENT and 'MEMORY' in the E segment line above. Note, however, that these segments are loaded last only because you imposed this control on them, not because of any inherent capability in the linker or assembler operations.

Segment Addresses

The 8088 must be able to address all segments in memory. Any 20-bit number can be addressed. The 8088 represents these numbers as two 16-bit numbers (for example, hex F:12). The F is a canonical frame address, and the 12 is the offset.

The canonical frame address is the largest frame address or segment address that can contain the segment. An offset is the segment's location, offset from the beginning of the canonical frame.

The linker recognizes a segment by its canonical frame address and its offset within the frame.

To convert the address F:12 to a 20-bit number, shift the frame address left four bits and add the offset as shown in this example:

$$\begin{array}{r} \text{F0} \\ + 12 \\ \hline \text{F:12} = \quad 102 \quad (20\text{-bit address}) \end{array}$$

How the Linker Assigns Addresses

To assign addresses to segments, the linker orders each segment by segment and class name. On the basis of the alignment and size of each segment (assuming the segments are contiguous), the linker assigns a frame address and an offset to each segment. This information is used for resolving relocatable references. The addresses start at 0:0.

Relocation Fixups

The linker performs relocation fixups (or resolves) on four types of references in object modules: Short, Near self-relative, Near segment-relative, and Long.

- **Short References:** Short references are all self-relative. The implication is that the frame address of the target and source frames are the same. The linker generates the fixup error message:

Fixup offset exceeds field width

under either of the following conditions:

- The target and source frame addresses are different.
- The target is more than 128 bytes before or after the source frame address.

The resulting value of the short reference must fit into one signed byte.

- **Near Self-Relative References:** When near self-relative references are used, the frame address of the target and source frames are the same. The linker generates the fixup error message under either of the following conditions:

- The target and source frame addresses are different.
- The target is more than 32K before or after the source frame address.

The resulting value of the near self-relative reference must fit into one signed word (16 bits).

- Near Segment-Relative References: Given the target's canonical frame, another frame is specified (via an ASSUME directive or the : operator in assembly language or via a high-level language convention). The target must be addressable through the canonical frame specified. The linker generates the fixup error message under either of the following conditions:
 - The offset of the target within the specified frame is greater than 64K or less than zero.
 - The beginning of the canonical frame of the target is not addressable by the specified frame.

The resulting value of a near segment-relative reference must be an unsigned word (16 bits).

- Long References: Long references have a target and another frame (specified by an ASSUME or by a high-level language). The target must be addressable through the canonical frame specified. The linker generates the fixup error message under either of the following conditions:
 - The offset of the target within the specified frame is greater than 64K or less than zero.
 - The beginning of the canonical frame of the target is not addressable by the specified frame.

The resulting value of a long reference must be a frame address and an offset.

MS-LIB The Library Manager

This utility is provided for advanced programmers. If you are not an advanced user, you have no need for this utility.

With the MS-LIB library manager, you can create library files to use with MS-LINK. 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.

MS-LIB requires at least 38K bytes of memory (28K bytes for code and 10K bytes for run space).

Order of Operations

During each library session, MS-LIB first deletes or extracts modules; then it appends new modules to the end of the file. During those operations, MS-LIB reads each module into memory and checks it for consistency. It then writes back to the file all modules you wish 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, MS-LIB creates the index that MS-LINK uses to find modules and symbols in the library file. If you wish, you can instruct MS-LIB to store the index in a listing 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.

Running MS-LIB

To start MS-LIB, you can do any of the following:

- Enter the LIB command without options; then respond to the three prompts.
- Enter the LIB command with options, thus avoiding the prompts.
- Enter the LIB command, specifying a response file (a file that contains answers to the prompts).

Several command characters help simplify the task of using MS-LIB. They are discussed later in the “Command Characters” section. You may want to refer to that section when reading about the different methods for starting MS-LIB.

Method 1: The Keyboard Responses

To use this method, type:

```
LIB 
```

This command loads MS-LIB into memory and displays three prompts, one at a time. The prompts are discussed in detail below.

Library File:

Enter the name of the library you want to create or modify. If you omit an extension, MS-LIB assumes .LIB. For example, to specify the library SAMPLE.LIB, at the prompt type:

```
SAMPLE 
```

If you specify a library that does not exist, MS-LIB displays:

```
Library file does not exist. Create?
```

Type YES to create the file or NO to stop the session.

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 (-), which deletes
- Asterisk (*), which extracts
- Plus sign (+), which appends

For example, to delete the module LESS and append the object file MORE, at the prompt type:

```
+MORE-LESS 
```

Default drive specifications and extensions for the `Operation:` prompt vary with the type of operation. See “Command Characters” for a detailed explanation of the command characters and defaults.

List file:

Enter the filename of the listing file you want to create. For example, to create the listing file *CROSSLST*, at the prompt, type:

```
CROSSLST 
```

If you press only at the prompt, MS-LIB uses NUL and thus does not create a listing file.

Method 2: Responses on Command Line

To use this method, enter the command in this format: LIB *library operations, listing*

The command line option, defined under “Method 1: Keyboard Responses,” 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 *library* and the first command character.

If you type only a library name followed by a semicolon (;) MS-LIB reads the library file and checks it for consistency. It performs no other operations. 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 listing filename, MS-LIB checks the library file for consistency and produces the listing file. It performs no other operations. For example, to perform a consistency check on the library file PASCAL and then create the listing file PASCROSS.PUB, type:

```
LIB PASCAL , PASCROSS.PUB 
```

Method 3: Response File

To use this method, enter the command in the form:

```
LIB @filespec 
```

filespec is the name of a previously created file that contains responses to the MS-LIB command prompts.

A response file has one line of text for each response. Be sure your responses are in order so that they apply to the appropriate prompts.

Use the command characters in the response file the same as you use command characters on the keyboard.

When the library session begins, MS-LIB displays each prompt with its response. Whenever you have not specified a response, MS-LIB uses the default.

If you type a library filename followed by a semicolon, MS-LIB reads the file and checks it for consistency but performs no other operations.

If you enter a library filename and then a comma, and then a listing filename, MS-LIB performs the consistency check and produces the listing file.

Here are two sample response files:

```
PASCAL   
,CROSSLST 
```

The above file causes MS-LIB to read the library file PASCAL.LIB, perform a consistency check, and create the listing file CROSSLST.

```
PASCAL   
+MORE-LESS 
```

The above file causes MS-LIB to delete the module LESS from the library file PASCAL.LIB and add the object file MORE.OBJ. MS-LIB does not create a listing file.

Command Characters

Several command characters help simplify the task of using MS-LIB. Four of these specify particular operations. They are:

- Plus sign (+) append
- Minus sign (-) delete
- Asterisk (*) extract
- Ampersand (&) extend line

The other command characters are the semicolon (;), which selects default responses, and , which stops the session.

Plus Sign (+)

A plus sign preceding an object filename instructs MS-LIB to append that object file as a module in the specified library.

When MS-LIB does this, it removes the drive specification and extension from the object file specification. For example, the object file B:CURSOR.OBJ becomes module CURSOR.

Minus Sign (-)

A minus sign preceding a module name instructs MS-LIB to delete that module.

Asterisk (*)

An asterisk preceding a module name instructs MS-LIB to extract (copy) that module from the library into an object file.

MS-LIB 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, copy the file to another. If you want to use the current drive but not the .OBJ extension, rename the file.

Ampersand (&)

The ampersand extends the current line when you specify the operations to perform. When you place an ampersand at the end of a line, MS-LIB 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.

Semicolon (;)

At any time after the first prompt (`Library File:`), you can select default responses to the remaining prompt(s). To do so type `;`

Caution: Once you have entered the semicolon, you can no longer respond to any of the remaining prompts. Therefore, do not use the semicolon to skip only the `Operation:` prompt. To skip one prompt, use the key.

Typing stops the library session at any time. If you enter an incorrect response, such as an incorrect filename or module name, type to exit MS-LIB. Then restart the session.

If you make an error before you press use to delete characters in that line.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

THE DEBUG UTILITY

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 without reassembling the program.

To enter the DEBUG program, type:

```
DEBUG [pathname] [parameters] ENTER
```

- *pathname* specifies the program file to be loaded 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 will cause 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 included a *pathname*.

DEBUG displays a hyphen (-) to indicate that it is ready to accept a command.

At initialization, the DEBUG segment registers (CS, DS, DX, 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, it does not run properly.

DEBUG Commands

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 (described in Chapter 1) and the MS-DOS editing functions (described in Part 3.)

Notes

- You can terminate any DEBUG command by typing **CTRL C**.
- To stop the screen from scrolling, type **CTRL S**; to continue scrolling, press the space bar.
- If a syntax error occurs in a DEBUG command, DEBUG displays the command line with the error indicated by an ↑ and the word error. For example:

```
D CS:000 CS:110
    ↑ Error
```

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
```

Command Summary

Command	Syntax
Assemble	A [<i>address</i>]
Compare	C <i>range address</i>
Dump	D [<i>range address</i>]
Enter	E <i>address [list]</i>
Fill	F <i>range list</i>
Go	G [= <i>address1 [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>range address</i>]
Write	W [<i>address [drive sector sectorcount]</i>]

Command Parameters

address is a one- or two-part designation in one of the following formats.

- An alphabetic segment register designation and an offset value. Example: CS:0100

- A 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 G, L, T, U, and W, for which the default segment is CS.

All numeric values are hexadecimal. A colon is required to separate a segment designation and an offset.

byte is a one- or two-character hexadecimal value placed in or read from an address or register.

drive is a one-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 should be specified.)

list is a series of strings or byte values. List must be the last command line parameter. For 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.

Example: CS:100 L 10
 CS:100

The limit for *range* is 10000 hex. To specify a value of 10000 hex within four digits, enter 0000 (or 0).

registername (See the explanation of the Register command for a list of valid register names.)

sector sectorcount is a one- to three-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.

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, the Assemble command 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

- 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. The NEAR prefix can be abbreviated to NE.

Word Ptr/Byte Ptr

```
NEG      BYTE PTR [128]
DEC      WD [SI]
```

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* will be 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.

Example

```
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 be displayed.

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 a period (.)

- Each displayed line begins on a 16-byte boundary and shows 16 bytes. A hyphen appears between the eighth and ninth bytes.
- If only a starting address is specified with the D command, the contents of memory are displayed starting at the address.
- If neither parameter is specified with the D command, 128 bytes are displayed beginning immediately after the last address displayed by a previous D command.
- If only an offset for the starting address is given, 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
```

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 the new values to be entered.

Notes

- 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 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 eight-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 time you press the hyphen, the DEBUG Enter command backs up one more byte without changing the current byte.
- Press **ENTER** to end the Enter command.

If you enter only an offset for the address, the segment indicated by register DS is used.

Example

```
E CS:100 ENTER
```

causes DEBUG to enter byte values into memory beginning at DS: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 the hyphen 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 **ENTER** to end the Enter command.

FILL

F range list

~ Fills the memory locations in the specified range with the values in the list.

Parameters

range is the area of memory to receive the new values.

list is the data values to be placed in *range*.

Notes

- If the list contains fewer bytes than the range, *DEBUG* uses the list until all locations in the range are filled.
- If the 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 the range, *DEBUG* uses the segment indicated by Register DS.

Example

```
F 04BA:100 L 100 42 45 52 54 41 
```

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 be executed are displayed.

Parameters

address1 is the start location of the program to be executed.

address2... gives the addresses for optional breakpoints.

Notes

- If you include *address1*, execution begins at *address1* in the CS segment. The equal sign (=) is required.
- If *address1* is omitted, the program execution starts at the current instruction. The address of the current instruction is determined from the contents of the CS and IP registers.

- *address2* is used to set breakpoints. These allow you to examine the register contents and flag settings in effect when a specified address is reached during program execution.

After execution halts at a breakpoint, you can enter the Go command again. The program resumes execution at the instruction after the breakpoint.

- 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.
- A maximum of ten addresses can be set, in any order. If you enter only an offset for a breakpoint address, the segment indicated by register CS is used. Breakpoints can be set only at addresses that contain the first byte of an 8088 opcode. Execution stops when any breakpoint is reached.
- The user stack pointer must be valid and have 6 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.

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, displays the register contents and the flags, and 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*. The sum and difference are displayed on one line.

Example

```
H 19F 10A [ENTER]
```

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, DEBUG displays:

```
42
```

LOAD

L [*address*[*drive sector sectorcount*]]

Loads a file into memory.

Parameters

address is the beginning memory location for a file load.

drive sector is the absolute disk sectors that contain the file to be loaded.

sectorcount is the number of disk sectors to be loaded.

Notes

- Before a file can be loaded, it must have been named either when DEBUG was started 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.
- When all parameters are specified, absolute disk sectors are loaded. The sectors are loaded 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* have been loaded.
- When an offset is used for the starting address, the segment indicated by register CS is used.

- 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. The header is stripped from the *.exe* file before it is loaded. Thus the size of an *.exe* file on disk differs from its size in memory.

If the file indicated by the Name command, or specified when DEBUG is started, is a *.hex* file, then entering the L command without parameters causes DEBUG to load the file at the address specified in the *.hex* file. If the L 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

range is the beginning location and number of bytes of memory to be moved.

address is the beginning address location to receive the moved data.

Notes

- Overlapping moves (where some locations in the range are also in the block beginning at address) are performed without loss of data. The addresses in the range remain unchanged unless new data is written to them by the move.
- If only an offset is used for the starting address of *range* or for *address*, the segment indicated by register DS is used. To specify an ending address for the range, enter only an offset value.

Example

```
M CS:100 110 CS:500 
```

causes DEBUG to move the data that is between CS:100 and CS:110 to the memory area beginning at CS:500

NAME

N *filespec1* [*filespec2*...]

Assigns filespecs or program names for later Load or Write commands. If DEBUG is entered without naming any file, then you must use the N *filespec* command before a file can be loaded. The Name command also assigns filespec parameters to the file being debugged.

Parameters

filespec1 filespec2 are names assigned to the files you will be using with DEBUG.

Notes

- Four areas of memory can be affected by the Name command:

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 filespec 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 File1.exe is loaded into memory, the Name command is used again this time to specify the parameters to be used by File1.exe. When the Go command is executed, File1.exe is executed as if file1 file2.dat file3.dat had been entered at the MS-DOS command level.

```
DEBUG prog.com 
N param1 param2/c 
G 
```

The Go command executes the file in memory as if `PROG param1 param2/C` had been typed at the MS-DOS command level.

OUTPUT

O *portaddress* *byte*

Sends the byte to the specified portaddress.

Parameters

portaddress *byte* allows a 16-bit port address.

Example

```
O 2F8 4F
```

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 very similar to Trace, except for the following functions:

- It automatically executes.
- It returns from any call or software interrupt instructions.
- It executes loop and repeat string instructions.

Parameters

address is the address of the instruction where 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.

Example

Q

ends DEBUG and returns to the MS-DOS system prompt.

REGISTER

R [*registername*]

Performs 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 to be displayed.

Notes

- If you enter R with no registername, DEBUG displays the contents of all registers and flags, together 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 one- to four-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

Both IP and PC refer to the Instruction Pointer.

- Entering F as the registername causes DEBUG to display a two-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

At the RF command, DEBUG displays 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 to make the changes and exit the command. Flags for which you did not type a new value remain unchanged.

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 [ENTER]
```

DEBUG makes the requested changes.

```
R AX [ENTER]
```

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 specified memory for a list of bytes.

Parameters

range is the start and end address in memory where search will be conducted.

list are the byte values to find.

Notes

- 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 no addresses are displayed, the list was not found.
- 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 
```

causes DEBUG to search the addresses from CS:100 to CS:110 for 41H. If two matches are found, 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 where trace is to begin.

value is the number of instructions to be traced (beginning with *address*).

Notes

- 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 in order to study the registers and flags for any instruction. Press the space bar to continue scrolling.

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 instructions to be disassembled.

range is the number of bytes to disassemble.

Notes

- The display of disassembled code looks like a listing for an assembly language source file. The U 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 U command with the range specified, DEBUG disassembles all bytes in the range.
- In all cases, DEBUG may 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 may include more bytes than expected.
- When only an offset is used for the starting address, the segment indicated by register CS is used.
- If you have altered some locations using DEBUG, you can enter the U command for the changed locations, view the new instructions, and use the disassembled code to edit the source file.

Example

```
U04BA:0100 L10 
```

causes DEBUG to disassemble 16 bytes beginning at address 04BA:0100 and displays information similar to the following:

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

WRITE

W [*address*[*drive sector sectorcount*]]

Writes the data being debugged to a disk file.

Parameters

address is the memory location of the data to be saved.

drive sector designates the disk drive to be used for the save.

sector designates the starting diskette sector for the save.

sectorcount indicates the number of sectors to be used in the save.

Notes

- If no parameters are given, the file save begins at CS:100. If only the address parameter is given, the W 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 may 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 has not changed.)
- The file must have been named either with the DEBUG start-up 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 none is specified.

- When all parameters are specified, 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 *sectorcount* have been written.
- If only an offset is entered 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, and then press **ENTER**.
- Be very careful when you write to absolute sectors on the disk. Data that was previously in these sectors is destroyed.

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 is displayed when the write is complete.

PROBLEMS AND ERROR MESSAGES

It is important that you understand error messages. They inform you of difficulties the computer is having in accomplishing the task it has been given. Knowing what the problem is will assist you in solving it most efficiently.

MS-DOS Error Messages

The two sources of error messages are MS-DOS and an application program. If you receive an error message other than a system error, it comes from an application program. Refer to that program's documentation for help.

System errors consist of two types: those that occur while MS-DOS is writing to a disk or to another system device (such as a printer) and command errors. As well, MS-DOS special programs (such as LINK, DEBUG, or EDLIN) have their own error structure. The errors that can be displayed by MS-DOS or its functions are described in the following order:

- Device Error Messages
- Command Error Messages
- EDLIN Error Messages
- Linker Error Messages
- DEBUG Error Messages

Device Error Messages

Device errors refer to problems with your computer and its peripherals, such as disk drives, modems and printers. If a Device Error occurs at any time during a command or program, MS-DOS returns an error message in the following format:

type error action device name
Abort, Ignore, Retry:

type refers to one of the following types of errors:

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

action is either reading or writing.

device is the equipment that is experiencing the difficulty, such as drive (disk drive) or PRN (printer).

name is the letter of the drive (if *device* is a drive) in which the error occurred.

For example, if MS-DOS finds data on the diskette in Drive B that it is unable to read, it displays:

```
Data error reading drive B
Abort, Retry, Ignore?
```

After displaying the error message, MS-DOS waits for you to enter one of the following responses:

- A Abort. Terminate the program that was in process and return to the MS-DOS system prompt.
- I Ignore. Ignore the error and continue the program or command. If continuing requires further reads to the same disk area, the error message is likely to recur.
- R Retry. Repeat the disk read. Give this response if you have been able to correct the error.

To recover, press R to try again. If that fails, press A to cancel the operation. Then, you can take steps to discover the cause of the error.

Following are descriptions of the device error types MS-DOS can display.

Bad call format

The device driver was passed an incorrect length request header. Contact the dealer from whom you purchased the device driver.

Bad command

MS-DOS issued an invalid command for the device driver. Try the command again. If it continues to fail, contact the dealer from whom you bought the device driver.

Bad unit

A device driver passed an invalid sub-unit number. Contact the dealer from whom you purchased the device driver.

Data

The data cannot be read or written correctly. You probably have a flawed disk.

Disk

MS-DOS has found an error it does not recognize.

FCB unavailable

FCB's might not be high enough.

General failure

A hardware error or the disk is not formatted.

Invalid disk change

A diskette was removed while files were open.

Lock violation

All or part of a source file is locked and cannot be read. Wait a few moments and try again.

No paper

The printer is out of paper.

Non-DOS disk

The disk is not in a format recognizable by MS-DOS.

Not ready

The device is not ready. Check to see that the drive latches are closed and that all devices are on and ready.

Read fault

The device cannot successfully read the data. You may have a flawed disk. Try to copy all files to another disk. If this fails, you probably have a hardware problem. Contact your dealer.

Sector not found

MS-DOS is requesting a sector number higher than the highest number on the disk.

Seek

The disk drive or hard disk cannot locate the proper track on the disk.

Sharing violation

A file is open in a sharing mode that does not allow the command to write to the file.

Write Fault

The device cannot successfully write the data. You may have a flawed disk. Try to copy all files to another disk. If this fails, you probably have a hardware problem. Contact your dealer.

Write protect

You tried to write to a write-protected diskette. Remove the write-protect tab and try again.

One other error message might be related to faulty disk read or write:

File allocation table bad for drive *drive*

This message means that the copy in memory of one of the allocation tables has pointers to nonexistent blocks. Possibly the disk was incorrectly formatted or not formatted before use. If this error persists, the disk is currently unusable and must be formatted prior to use.

Command Error Messages

The following errors can occur while using MS-DOS commands.

Access denied

You tried to replace or copy to a write-protected, read-only, or locked file.

After format, one of system sectors could not be read

Errors were found on the system sectors of the disk. The disk might be unusable.

All files canceled by operator

This is a reminder only. You used the /T parameter with PRINT to cancel the printing of all files in the print queue.

Allocation error, size adjusted

A filename is displayed with this message. The File Allocation Table (FAT) contains an invalid sector number. CHKDSK automatically truncates the file at the end of the last valid sector number.

Bad command or file name

The command you typed is not valid. Check spelling and reenter the command. If the command is an external command, be sure it is in the directory that MS-DOS is searching for commands. If you are trying to execute a batch file, be sure you are in the directory which contains that file.

Bad or missing Command Interpreter

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)

You specified an invalid device in the CONFIG.SYS file. Check the accuracy of the DEVICE command in CONFIG.SYS.

Bad Partition Table

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

You used an invalid switch syntax with a command. Check the command reference for valid switches.

Cannot CHDIR to *filename*

Tree past this point not processed

CHKDSK is traveling the tree structure of the directory and is unable to go to the specified directory. No subdirectories below this directory are verified.

Cannot CHDIR to root

Processing cannot continue

The disk you are checking is bad. Restart the system and try using RECOVER command.

Cannot perform a cyclic copy

When using the /S switch, you cannot specify a target that is a subdirectory of the source.

Compare error(s) on

Track *t*, side *s*

Your source and target diskettes are not identical. DISKCOMP found that the data on the track indicated by *t* and the sector indicated by *s* does not match between the two diskettes.

Content of destination lost before copy

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)?

If you press (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 (for no) in response to this prompt, CHKDSK frees the lost blocks so they can be reallocated.

Destination disk format error

An error occurred during formatting. The disk may be flawed, or you may have inserted it in the drive incorrectly.

Directory is joined

CHKDSK cannot process directories that are joined.

Disk error reading FAT

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, use the COPY command to copy all files to another disk.

Disk error writing FAT

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, use the COPY command to copy all files to another disk.

Divide overflow

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)?

The target directory does not exist.

Drive not ready

The drive was not ready when PRINT attempted a disk access. PRINT keeps trying until the drive is ready. Be sure the disk that contains the file you are printing is in the disk drive.

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

CHKDSK does not write its corrections to disk unless you specify the /F switch. Specify the /F switch to correct the errors.

Error writing boot sector to destination

All floppy disks—both data and system disks—must have a boot sector so that they may be used by the system. The disk is either defective or needs to be reformatted.

Errors writing to the system cannot continue

Make sure you have a diskette in the drive specified by `FORMAT` and the drive door is closed. Make sure any write-protect tab is removed from the diskette to `FORMAT`. If the error persists, your diskette is unusable.

EXEC failure

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.

file canceled by operator

This is a reminder only. You used the `/C` parameter with `PRINT` to cancel the current file in the print queue.

File cannot be converted

The source file is not in the correct format. `CS:IP` does not meet either criterion discussed in the `EXE2BIN` command. `CS:IP` meets the *.com file* criterion but has segment fixups. The file is not a valid executable file.

File cannot be copied onto itself

You tried to copy a file to a file that has the same name. Either change the name of your copy, or put it in a different directory or on a different disk.

File creation error

You tried unsuccessfully to add a file to the directory. You either exceeded the files per directory, or the file already exists and is perhaps a read-only file. Run `CHKDSK` to determine if some other condition caused the error.

File not found

The specified file does not exist in the specified directory or the current directory. Use the `DIR` command to see which directory the file is in. Also check the spelling of the filename.

`filename` is cross-linked on cluster

This CHKDSK message names each of the two files in error. Copy both files, and delete both cross-linked files (the original files).

First cluster number is invalid
entry truncated

This error is corrected automatically by CHKDSK.

Fixups needed - base segment (hex):

The source (`.exe`) file contains information which indicates that a load segment is required for the file. Specify the absolute segment address at which the finished module is to be located.

Has invalid cluster, file truncated

The file contains an invalid pointer to the data area. CHKDSK corrects this error automatically by truncating the file to the last valid data block.

Incompatible system size

The hidden system files do not take up the same amount of space on the target disk as the new system requires.

Incorrect DOS version

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.

Insufficient memory

There is not enough available memory to run the command. If you can eliminate some files from memory, try again.

Insufficient room in root directory

Erase files in root and repeat CHKDSK

The command cannot process until you delete files in the root directory.

Insufficient disk space

The diskette in the specified drive does not have sufficient room to complete the operation. Replace the diskette with one having the required storage space.

Intermediate file error during pipe

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.

Invalid COMMAND.COM

Insert COMMAND.COM disk in default drive
and strike any key when ready

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

The country number you specified in CONFIG.SYS is not valid.

Invalid current directory

Processing cannot continue. Restart the system and rerun CHKDSK.

Invalid date

Enter new date:

Enter a valid date, making sure you use hyphens (-) or slashes (/) to separate the parts of the date.

Invalid directory

The specified directory does not exist on the current or specified diskette or you have specified an illegal directory name.

Invalid drive specification

Use a valid drive specification, making sure you include the colon (:).

Invalid number of parameters

Reenter the command, including the correct number of parameters.

Invalid parameter

You specified a parameter that does not exist. Check the command syntax in this manual.

Invalid path, not directory,
or directory not empty

An attempt to remove a directory failed. The directory might not exist, or it might not be empty. A directory **must** be empty before you can remove it.

Invalid time
Enter new time:

Check the command syntax and parameters in this manual. Then enter a valid time, making sure you use a colon (:) to separate hours, minutes, and seconds and a period (.) to separate hundredths of a second.

Label not found

The specified label does not exist in the batch file.

List output is not assigned to a device

The device specified as the PRINT output device is invalid.

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

If you type Y , CHKDSK creates a directory entry and a file in which you can resolve this problem (files created by CHKDSK are named FILEnnnn.CHK). CHKDSK displays: X bytes disk space freed. If you have not specified the /F switch, CHKDSK frees the clusters and displays: X bytes disk space would be freed.

Memory allocation error.
Cannot load MS-DOS, system halted

Restart MS-DOS. If the error persists, use a new copy of the MS-DOS system diskette.

No files match pathname

The files you tried to add to the queue do not exist.

No free file handles.
Cannot start COMMAND.COM, exiting

Restart MS-DOS. If the message persists, increase the value given for FILES in the CONFIG.SYS file.

No room for system on destination disk

If the target disk contains files you don't need, use the ERASE command to delete as many as necessary to make room for the system. Otherwise, use a different disk as the target.

Out of environment space

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.

Path too long

The pathname you specified is too long. You might have to change to a lower subdirectory to replace files in deep subdirectories.

PRINT queue is empty

There are no files in the print queue.

PRINT queue is full

The queue can contain no more than ten files.

Probable non-DOS disk

Continue (Y/N)?

You are not using an MS-DOS disk. Type Y to continue processing or N to stop processing.

Read error in pathname

FC could not read the entire file.

Sector size too large in file (filename)

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.

Specified MS-DOS search directory bad

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.

Syntax error

Reenter the command, using the proper syntax.

Unable to create a directory

MS-DOS cannot create the directory you specified. There might be a name conflict (you have another file with the same name), or the disk might be full.

Unrecoverable error in directory
Convert directory to file (Y/N)?

If you type *Y* , CHKDSK converts the bad directory into a file. You can then fix the directory or delete it. If you type *N* , the directory becomes unusable, and you can neither fix nor delete it. You should always type *Y* .

Use HFORMAT to format hard disks

You are trying to use the 2.11 FORMAT command to format a hard disk. Use HFORMAT instead.

Warning-directory full

The disk does not contain enough directory space to recover more files. Copy some files to another disk, then erase the originals. Run RECOVER again.

WARNING -Read error on EXE file

The amount read was less than size in header. This is a warning message only.

EDLIN Error Messages

When you enter an invalid EDLIN command or fail to follow the proper syntax, EDLIN displays one of the following error messages:

Cannot edit .BAK file _ rename file

You are trying to edit a file with an extension of *.bak*. You cannot do so because EDLIN reserves the *.bak* extension for backup copies.

If you need the *.bak* file for editing purposes, you must either RENAME the file with a different extension or COPY it, giving it a different extension.

Disk full _ write not completed

You entered the End command, but the disk does not contain enough free space for the whole file. EDLIN terminated the End command and returned you to the operating system. Some of the file may have been written to the disk.

Only part (if any) of the file is saved. Delete that part of the file and restart the editing session. The file is not available after this error. Before you begin any editing session, be sure the disk has sufficient free space for the file.

Entry Error

The last command typed contained a syntax error. Reenter the command, using the correct syntax.

File Creation Error

The EDLIN temporary file cannot be created.

Verify that the directory has enough space to create the temporary file. Also be sure that the file does not have the same name as a subdirectory in the directory where the file to be edited is located.

File name must be specified

You did not specify a filename when you started EDLIN. Specify a filename whenever you start EDLIN.

File not found

The filename specified during a Transfer command was not found. Specify a valid filename when issuing a Transfer command.

Insufficient memory

There is not enough memory to run EDLIN.

Free some memory by writing files to disk or by deleting files before restarting EDLIN. Use the Write command and then the Append command.

Invalid drive name or file

You specified an illegal drive or filename when you started EDLIN. Specify a legal drive or filename.

Invalid Parameter

You specified a switch other than /B when starting EDLIN.
Specify the /B switch when you start EDLIN.

Line too long

During a Replace command, the string given as the replacement caused the line to expand beyond the limit of 253 characters. EDLIN terminated the Replace command.

Divide the long line into two lines, and then try the Replace command again.

Must specify destination number

You did not specify a line number in a Copy or a Move command. Renter the command, including a destination line number.

Not enough room to merge the entire file

There is not enough room in memory to hold the file during a Transfer command.

Free some memory by writing some files to disk or by deleting some files before you can transfer the file.

No room in directory for file

You are trying to create a file, but either the file directory is full or you are specifying an illegal disk drive or filename.

Check the command line that started EDLIN for either an illegal filename or an illegal drive specification. If the command is no longer on the screen, and if you have not yet typed a new command, you can recover the command by pressing the Copy All key, **F3**.

If the command line contains no illegal entries, run the CHKDSK program for the specified disk drive. If the status report shows that the disk directory is full, remove the disk. Insert and format a new disk.

Linker Error Messages

All messages, except for the warning messages, cause the link session to end. After you locate and correct a problem, you must restart the linker.

Messages appear in the list file and are displayed, unless you direct the list file to CON. If you direct the file to CON, the error messages are suppressed.

Attempt to access data outside of segment bounds,
possibly bad object module

This usually indicates that a bad object file exists.

Bad numeric parameter

A numeric value is not given as digits.

Cannot open temporary file

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 List map file.

Error: dup record too complex

A DUP record in an assembly language module is too complex. Simplify the DUP record.

Error: fixup offset exceeds field width

An assembly language instruction refers to an address with a short instruction instead of a long instruction. Edit the assembly language source and reassemble.

Input file read error

This usually indicates that a bad object file exists.

Invalid object module

An object module or modules are incorrectly formed or incomplete (as when assembly is stopped in the middle).

Symbol defined more than once

The linker found two or more modules that define one symbol name.

Program size or number of segments exceeds capacity of linker

The total size cannot exceed 384K bytes, and the number of segments cannot exceed 255.

Requested stack size exceeds 64K

Specify a size less than or equal to 64K bytes with the /STACK switch.

Segment size exceeds 64K

The addressing system limit is 64K bytes.

Symbol table capacity exceeded

The number or length of the names caused them to exceed the limit of approximately 25K bytes.

Too many external symbols in one module

The limit is 256 external symbols per module.

Too many groups

The limit is ten groups.

Too many libraries specified

The limit is eight libraries.

Too many public symbols

The limit is 1024 public symbols.

Too many segments or classes

The limit is 256 (segments and classes taken together).

Unresolved externals: list

The external symbols listed have no defining module among the modules or library files specified.

VM read error

This is a disk error; it is not caused by the linker.

Warning: no stack segment

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

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).

Write error in tmp file

No disk space remains in which to expand the *VM.TMP* file.

Write error on run file

This usually indicates there is not enough disk space for the run file.

DEBUG Error Messages

DEBUG can display any of the following error messages. Each error ends the DEBUG command under which it occurred, but does not end DEBUG itself.

BF Error (Bad flag)

The characters entered to alter flag were not allowable flag values. See the explanation of the Register command for the list of valid flag entries.

BP Error (Too many breakpoints)

More than ten breakpoints were included in the Go command. Retype the Go command again with ten or fewer breakpoints.

BR Error (Bad register)

The Register command has been given an invalid register name. See the Register command for the list of valid register names.

DF Error (Double flag)

Two values were entered for one flag when using the Register command. You may specify a flag value only once per RF command.

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-5AH (hexadecimal) that uniquely describes which key is pressed.
- KEYBOARD LEGEND — the physical marking(s) on the key. If multiple markings exist, they are listed from top to bottom.
- NORMAL — the normal (unshifted) ASCII value (returned when only the indicated key is pressed).
- SHIFT — the shifted ASCII value (returned when SHIFT is also pressed).
- CTRL — the control ASCII value (returned when CTRL is also pressed).
- ALT — the alternate ASCII value (returned when ALT is also pressed).
- REMARK — any remarks or special functions.

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. Double asterisks (**) indicate that no ASCII code is generated and that, instead, the special function described in the REMARK column is performed.

Appendix A / ASCII and Scan Codes

SCAN CODE	KEYBOARD LEGEND	ASCII CODES				REMARK
		NORMAL	SHIFT	CTRL	ALT	
01	ESC	01B	01B	01B	—	
02	! 1	031	021	—	X078	
03	@ 2	032	040	X000	X079	
04	# 3	033	023	—	X07A	
05	\$ 4	034	024	—	X07B	
06	% 5	035	025	—	X07C	
07	^ 6	036	05E	01E	X07D	
08	& 7	037	026	—	X07E	
09	* 8	038	02A	—	X07F	
0A	(9	039	028	—	X080	
0B) 0	030	029	—	X081	
0C	—	02D	05F	01F	X082	
0D	+ =	03D	02B	—	X083	
0E	BACK SPACE	008	008	07F	X08C	
0F	TAB	009	X00F	X08D	X08E	
10	Q	071	051	011	X010	
11	W	077	057	017	X011	
12	E	065	045	005	X012	
13	R	072	052	012	X013	
14	T	074	054	014	X014	
15	Y	079	059	019	X015	
16	U	075	055	015	X016	
17	I	069	049	009	X017	
18	O	06F	04F	00F	X018	
19	P	070	050	010	X019	
1A	[{	05B	07B	01B	—	
1B] }	a05D	07D	01D	—	
1C	ENTER	00D	00D	00A	X08F	(mail keyboard)
1D	CTRL	*	—	—	—	control mode
1E	A	061	041	001	X01E	
1F	S	073	053	013	X01F	
20	D	064	044	004	X020	
21	F	066	046	006	X021	
22	G	067	047	007	X022	
23	H	068	048	008	X023	
24	J	06A	04A	00A	X024	
25	K	06B	04B	00B	X025	
26	L	06C	04C	00C	X026	
27	; :	03B	03A	—	—	
28	' "	027	022	—	—	
29		X048	X085	X09	X091	
2A	LEFT SHIFT	*	—	—	—	SHIFT
2B		X04B	X087	X073	X092	
2C	Z	07A	05A	01A	X02C	
2D	X	078	058	018	X02D	
2E	C	063	043	003	X02E	
2F	V	076	056	016	X02F	
30	B	0a62	042	002	X030	
31	N	06E	04E	00E	X031	
32	M	06D	04D	00D	X032	
33	, <	02C	03C	—	—	
34	. >	02E	03E	—	—	

Appendix A / ASCII and Scan Codes

SCAN CODE	KEYBOARD LEGEND	ASCII CODES				REMARK
		NORMAL	SHIFT	CTRL	ALT	
35	/ ?	02F	03F	—	—	
36	RIGHT SHIFT	*	—	—	—	SHIFT
37	PRINT SCREEN	*	*	X072	X046	print screen
38	ALT	*	—	—	—	alternate mode
39	SPACE BAR	020	020	020	x020	
3A	CAPS LOCK	*	—	—	—	caps lock
3B	F1	X03B	X054	X05E	X068	
3C	F2	X03C	X055	X05F	X069	
3D	F3	X03D	X056	X060	X06A	
3E	F4	X03E	X057	X061	X06B	
3F	F5	X03F	X058	X062	X06C	
40	F6	X040	X059	X063	X06D	
41	F7	X041	X05A	X064	X06E	
42	F8	X042	X05B	X065	X06F	
43	F9	X043	X05C	X066	X070	
44	F10	X044	X05D	X067	X071	
45	NUM LOCK	*	—	—	—	number lock
46	HOLD	*	*	*	*	freeze display
47	7 \	037	05C	X093	†	
48	8 `	038	07E	X094	†	
49	9 pG UP	039	X049	X084	†	
4A		X050	X086	X096	X097	
4B	4	034	07C	X095		
4C	5	035	—	—	†	
4D	6	036	—	—	†	
4E		X04D	X088	074	—	
4F	END 1	031	X04F	X075	—	
50	2	032	060	X09A	†	
51	3 PG DN	033	X051	X076	†	
52	0	030	X09B	X09C	†	
53	- DELTE	02D	X053	X09D	X09E	
54	BREAK	000	000	*	X400	control break routine (INT 1BH)
55	+ INSERT	02B	052	X09F	X0A0	
56	.	02E	X0A1	X0A4	X0A5	(numeric keypad)
57	ENTER	00D	00D	00A	X08F	(numeric keypad)
58	HOME	X047	X04A	X077	X0A6	
59	F11	X098	X0A2	X0AC	X0B6	
5A	F12	X099	X0A3	X0AD	X0B7	

* Indicates special functions performed

- means this key combination is suppressed in the keyboard driver
- X values preceded by X are extended ASCII codes (codes preceded by an ASCII NUL)
- † The `[ALT]` key provides a way to generate the ASCII codes of decimal numbers between 1 and 255. Hold down the `[ALT]` key while you type *on the numeric keypad* any decimal number between 1 and 255. When you release ALT, the ASCII code of the number typed is generated and displayed.

Note: When the NUM LOCK light is off, the Normal and SHIFT columns for these keys should be reversed.

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 may 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	0	047	2FH
DLE	016	10H	1	048	30H
DC1	017	11H	2	049	31H
DC2	018	12H	3	050	32H
DC3	019	13H	4	051	33H
DC4	020	14H	5	052	34H
NAK	021	15H	6	053	35H
SYN	022	16H	7	054	36H
ETB	023	17H	8	055	37H
CAN	024	18	9	056	38H
EM	025	H19H	:	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 A / 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)	069	E
033	!	070	F
034	''	071	G
035	#	072	H
036	\$	073	I
037	%	074	J
038	&	075	K
039	'	076	L
040	(077	M
041)	078	N
042	*	079	O
043	+	080	P
044	,	081	Q
045	-	082	R
046	.	083	S
047	/	084	T
048	0	085	U
049	1	086	V
050	2	087	W
051	3	088	X
052	4	089	Y
053	5	090	Z
054	6	091	[
055	7	092	\
056	8	093]
057	9	094	^
058	:	095	_
059	;	096	:
060	<	097	a
061	=	098	b
062	>	099	c
063	?	100	d
064	@	101	e
065	A	102	f
066	B	103	g
067	C	104	h
068	D	105	i

ASCII CHARACTER CODES

ASCII Code	Character	ASCII Code	Character
106	j	143	Ë
107	k	144	È
108	l	145	æ
109	m	146	Æ
110	n	147	ô
111	o	148	Ö
112	p	149	ò
113	q	150	Ô
114	r	151	ù
115	s	152	ÿ
116	t	153	Ö
117	u	154	Ü
118	v	155	ç
119	w	156	£
120	x	157	¥
121	y	158	Pt
122	z	159	f
123	{	160	á
124		161	í
125	}	162	ó
126	~	163	ú
127	☐	164	ñ
128	Ç	165	Ñ
129	ü	166	ä
130	é	167	ö
131	â	168	¿
132	ä	169	┌
133	·ä	170	┐
134	ã	171	½
135	ç	172	¼
136	ê	173	ì
137	ë	174	«
138	è	175	»
139	ï	176	▩
140	î	177	▩
141	ì	178	▩
142	Ä	179	

ASCII CHARACTER CODES

ASCII Code	Character	ASCII Code	Character
180	┌	218	┐
181	┐	219	■
182	┌┐	220	▬
183	┐┌	221	▬
184	┐┐	222	▬
185	┐┐	223	▬
186	▯	224	α
187	▮	225	β
188	▮	226	┌
189	▮	227	π
190	▮	228	Σ
191	┌┐	229	σ
192	┌	230	μ
193	┐	231	τ
194	┐	232	⊖
195	┐	233	⊖
196	┐	234	Ω
197	┐	235	δ
198	┐	236	∞
199	┐	237	∅
200	┐	238	€
201	┐	239	∩
202	┐	240	≡
203	┐	241	±
204	┐	242	≥
205	┐	243	≤
206	┐	244	ƒ
207	┐	245	J
208	┐	246	÷
209	┐	247	≈
210	┐	248	°
211	┐	249	●
212	┐	250	•
213	┐	251	√
214	┐	252	η
215	┐	253	²
216	┐	254	■
217	┐	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 startup 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 with the COPY command:

```
copy con config.sys <ENTER>
```

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 **F6** 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. (See Appendix C 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 <i>disk buffers</i> .
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 [*switch*]

Sets the **CTRL** **C** check. *Switch* can be ON or OFF.

Normally, MS-DOS checks for **CTRL** **C** only while writing to the screen or printer, reading from the keyboard, or performing asynchronous communications. Therefore, 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 table lists the valid country codes. The default is 001 (for the United States).

Country	Code	Country	Code
Australia	061	Middle East	785
Belgium	032	Netherlands	031
French Canada	002	Norway	047
Denmark	045	Portugal	351
Finland	358	Spain	034
France	033	Sweden	046
Germany	049	Switzerland	041
Italy	039	United Kingdom	044
Israel	972	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 C for more information on the device driver commands. If you have written your own device drivers, you can 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, place 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:*n* [/C] [/F:*n*] [/H:*n*] [/N] [/S:*n*]
[/T]

Lets you define parameters for block devices when you start MS-DOS, overriding the original MS-DOS device driver settings.

/D:*n* specifies a logical drive number. *n* can be in the range 0 to 255. Drive A = 0, Drive B = 1, Drive C = 2, and so on.

/C specified that *changeline* (doorlock) support is required.

/F:n specifies the form factor index. *n* can be:

- 0 = 320/360K floppy
- 1 = 1.2 M 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:n specifies the maximum head number. *n* is a value in the range 1 to 99.

/N tells MS-DOS that the device is non-removable.

/S:n specifies the number of sectors per track. *n* is a value in the range 1 to 99.

/T:n specifies the number of tracks per side. *n* 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 /F:2
```

or

```
DRIVPARM=/D:1
```

Defines Drive B as a 3½-inch floppy disk drive. Note that you can omit the /F parameter, which defaults to 2 for 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 maximum number of files that MS-DOS can access at one time. The default is ten files. Allocating more files than required by your system or application decreases available memory.

number is the number of open files that the system calls can access. (System calls in the range 2FH to 60H are compatible with the XENIX operating system.) The maximum *number* is 99.

Example:

```
files=20
```

sets the number of accessible open file 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. Therefore, 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 C, "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 top-level command processor.

pathname specifies the command processor to substitute as the top-level 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=12
device=\bin\network.sys
break=on
shell=a:\bin\command.com /e:3000 a:\bin /p
lastdrive=e
```

This file sets the number of disk buffers to 10 and the maximum number of open files to 12. 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 C for more information on DRIVER.SYS.)

Changing CONFIG.SYS

If CONFIG.SYS is already on your diskette, before creating a new file be sure it does not contain data you want. To read an existing file, type:

```
type config.sys 
```

CONFIG.SYS is listed to the screen. If you wish to create a new CONFIG.SYS file, but want to be able to re-establish the same configuration later, rename the existing CONFIG.SYS file. For example, type:

```
ren config.sys config.old 
```

You now have a file named CONFIG.OLD on your diskette. You can later change the name back to CONFIG.SYS to re-establish the same configuration. To do so, first delete or rename the current CONFIG.SYS file, then type:

```
ren config.old to config.sys 
```

To expand an existing CONFIG.SYS file to add a new device driver, you might type:

```
copy config.sys+con   
device=myprint.prt   
 
```

The command DEVICE = MYPRINT is appended to the current CONFIG.SYS file.

To create a totally new CONFIG.SYS file with 20 buffers, BREAK set to ON, and a new printer driver, you might type:

```
copy con config.sys   
buffers = 20   
break = on   
device = myprint.prt   
 
```

You can create a new CONFIG.SYS file at anytime with such a procedure. The new file you create overwrites any old CONFIG.SYS file. Be sure the CONFIG.SYS file you create is in the root directory.

See the COPY command in Chapter 6 and Chapter 4, "Batch Files," for more information on using COPY and CON.

INSTALLABLE DEVICE DRIVERS

Your MS-DOS system can use several *device drivers*. A device driver is a program that controls external devices or executes other programs. The Version 2.11 MS-DOS device drivers are included on the MS-DOS/BASIC system diskette. Version 2.11 drivers are: ANSI.SYS, KEYCVRT.SYS, LPDRVR.SYS, SPOOLER.SYS, and VDISK.SYS. In Version 3.2 MS-DOS, both the MS-DOS/BASIC system diskette and the Supplemental Programs diskette contain device drivers. Version 3.2 drivers are: ANSI.SYS, DRIVER.SYS, KEYCVRT.SYS, LPDRVR.SYS, MLPART.SYS, SPOOLER.SYS, and VDISK.SYS.

Device Driver	Purpose
ANSI.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.
KEYCVRT.SYS	Converts certain keyboard characters for application software compatibility.
LPDRVR.SYS	Lets you configure your system to take full advantage of your printer's capabilities.
MLPART.SYS	Lets you access multiple MS-DOS partitions on a hard disk drive.
SPOOLER.SYS	Lets you continue processing data while printing.
VDISK.SYS	Lets you establish a <i>virtual disk drive</i> in RAM.

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 *Programmer's Reference Manual*.

Include the following command in your CONFIG.SYS file to install the ANSI.SYS driver:

```
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.

To install the DRIVER.SYS driver, include a DEVICE command in your CONFIG.SYS file, using this format:

```
DEVICE = DRIVER.SYS /D:n [/C] [/F:n] [/H:n] [/N] [/S:n]
[/T:n]
```

where:

/D:n 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. (The first external floppy drive is always hexadecimal 02.)

If you have one floppy disk drive and one hard disk drive, the floppy disk drive is hexadecimal 00, and the hard disk drive is hexadecimal 80.

/C specifies that *changeline* (doorlock) support is required. This is a hardware feature that lets the software know if the drive door is open.

/F:*n* specifies the *form factor index*, where *n* can be:

- 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:*n* specifies the maximum head number. *n* is a value in the range 1 to 99.

/N tells MS-DOS that the device is non-removable.

/S:*n* specifies the number of sectors per track. *n* is a value in the range 1 to 99.

/T:*n* specifies the number of tracks per side. *n* is a value in the range 1 to 999.

KEYCNVRT.SYS

(For Key Conversion)

KEYCNVRT.SYS lets you generate keyboard characters other than Tandy scan codes. This driver is for application software compatibility.

Include the following DEVICE command in your CONFIG.SYS file to install the KEYCNVRT.SYS driver:

DEVICE = KEYCNVRT.SYS

Whenever you boot the system, KEYCNVRT.SYS converts the following keyboard codes:

Key	Tandy Scan ASCII Code	Converted To
Slash (/)	475C	2B5C
Vert. Line ()	4B7C	2B7C
Tilde (~)	487E	297E
Single Left quote (‘)	5060	2960
minus (-) (DELETE NUMLOCK)	532D	4A2D
plus (+) (INSERT NUMLOCK)	552B	4E2B

LPDRVR.SYS

(For Extended Printer Capabilities)

The loadable printer driver extends the useable capabilities of a Tandy printer. 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

Include this DEVICE command in your CONFIG.SYS file to install the printer driver:

DEVICE = LPDRVR.SYS

Next, 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 A.

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.

Finally, send the control code sequence, in ASCII form, to the printer driver. You can do this in any of three ways.

- Use BASIC's LPRINT statement with the CHR\$ function, as described in the *BASIC Reference Manual*.
- Make an MS-DOS function call, as described in the *MS-DOS Programmer's Reference Manual*.
- Make a BIOS call, as described in the *MS-DOS Programmer's Reference Manual*.

Example:

This example uses BASIC to set the number of lines per page to 55:

```
lprint chr$(27);chr$(67);chr$(55) [ENTER]
```

CHR\$(27) sends the ESCAPE, CHR\$(67) sends the C, and CHR\$(55) sends the number of lines.

Printer Control Codes

Function	Code	Result
Set lines per page	ESCAPE C;n;	Sets the page length to <i>n</i> lines. <i>n</i> is a number in the range 1-127. Issue this command before setting vertical tabs or form feed.
Set horizontal tabs	ESCAPE D;n1;n2 n3;...nk;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 1-80 in regular print mode or the range 1-132 in compressed print mode. When the printer is turned on, the tab stops are automatically set to every eight columns. Use ESCAPE D to change them.

Appendix C / Installable Device Drivers

Function	Code	Result
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 1-64. 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.
Horizontal Tab	HT	Tabs to the next horizontal tab stop.
Vertical Tab	VT	Tabs down to the next vertical tab stop.
Advance to top of page (form feed)	FF	Advances paper to the next top of page. When the printer is turned on, the top of page is automatically set to 66 lines from the printer position. To change the number of lines per page use ESCAPE C.
Skip perforation	ESCAPE N; <i>n</i> ;	Sets the number of lines to skip after printing each page to <i>n</i> . <i>n</i> is the number each time you change page length.
Cancel skip perforation	ESCAPE O	Cancels ESCAPE N.
Set 132 characters per line	SI	Turns on the compressed character mode.
Set 80 characters per line	DC2	Turns off the compressed character mode.

Function	Code	Result
Pass <i>n</i> codes directly to the printer	ESCAPE V; <i>n</i> ;	
Reset (cancel) driver	CAN or DEL	Resets all counters and tab stops to their default values.
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.

Converting Printer Code

LPDRVR also lets you convert any printer code to any other code before it is sent to the printer. It does this by maintaining a 256-byte table called a *character translation table*.

Initially code is translated to its original form but, by making changes to the table, you can set any code to be converted. To obtain the memory address of the code in the table, make the necessary BIOS call from an assembly-language program as follows:

1. Set the AH register to 3.
2. Execute an INT 17H instruction.

The pointer at the beginning of the character translation table is returned in the ES:BX registers. Each code is offset from the memory location pointed to by ES:BX by its ASCII value. 0FFH in the table indicates printer code 0FFH or indicates that a string translation is defined for that character. 0FFH can then be changed to redefine the character.

LPDRVR also lets you convert any printer code into a string of as many as eight codes before it is sent to the printer. You can define as many as 50 different replacement strings. This is done by sending the following escape code and code sequence to the printer driver:

```
esc wn code; string
```

where *n* is the length of the string that replaces the original code plus one. *code* is the original code. *string* is the list of replacement characters.

The MODE command supports some commonly used character set translations. For example, to enter the code to translate a 175 code to the characters ">>," type the following from BASIC:

```
lprint chr$(27);"W";chr$(3);chr$(175);">>";
```

Now, whenever you type:

```
lprint chr$(175);
```

the characters >> are printed. If the printed replacement string is wider than one character, horizontal tabs on the remainder of the printed line will be incorrect.



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.

Before installing the device driver, create the DOS2 partitions, using the MLPART command.

To install the MLPART.SYS device driver, include a DEVICE command in your CONFIG.SYS file, using this format:

```
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.

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.

To install the printer buffer driver, include a DEVICE command in your CONFIG.SYS file, using this format:

```
DEVICE = SPOOLER [/printer] [size]
```

where:

/printer is the number of the printer you want to use (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.

size is the size of the buffer in kilobytes. If you omit *size*, the system uses 20K.

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.
- 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.

To install the virtual disk driver, include a DEVICE command in your CONFIG.SYS file, using this format:

DEVICE = VDISK.SYS [*storage sector-size dir*]

storage specifies the size (in kilobytes) for the virtual disk. *storage* 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.

HARD DISK SETUP

The procedure for setting up your hard disk varies according to which version of the MS-DOS operating system you are using. Follow the steps outlined below to set up your hard disk for the version of the MS-DOS operating system you are using:

MS-DOS Version 2.11

All of the MS-DOS 2.11 hard disk setup programs are provided on the Hard Disk Utilities diskette that comes with the Hard Disk Controller.

1. Use the HSECT program to *hard format* the hard disk.
2. Use the FDISK program to partition the hard disk for MS-DOS. (Note that multiple MS-DOS partitions are not allowed.)
3. Use the HFORMAT program to *soft format* the MS-DOS partition and make it bootable. Use the /B switch to enter bad track information.

MS-DOS Version 3.2

Most of the MS-DOS 3.2 hard disk setup programs are provided on the Supplemental Programs diskette. The FORMAT program is provided on the MS-DOS/BASIC system diskette.

Note: Do not use the hard disk setup programs on the Hard Disk Utilities diskette with Version 3.2 MS-DOS. You do not need the Hard Disk Utilities diskette at all if you are using the MS-DOS Version 3.2 operating system.

1. Use the HSECT program to *hard format* the hard disk. HSECT prompts for the drive to be formatted.
2. Use the FDISK program to partition the hard disk for MS-DOS. Multiple MS-DOS partitions are allowed.
3. Use the FORMAT program to *soft format* the DOS partition and make it bootable. (Use the /S switch to copy the hidden system files to the hard disk.)

4. Copy all the files from your MS-DOS system diskette to your hard disk drive. If you are setting up only one hard disk partition, this is the final step.
5. Use the MLPART command (MLPART.COM) to create one or more non-bootable, DOS2 partitions on your hard disk.
6. Copy MLPART.SYS from the Supplemental Programs diskette to your hard disk drive. (Be sure to include a DEVICE=MLPART.SYS command in your hard disk's CONFIG.SYS file.)
7. Reboot the computer from the hard disk drive to install the MLPART.SYS device driver (load it into memory).
8. Use MLFORMAT to format any DOS2 partitions you created. (This is an optional step. Do this only if you use MLPART to create one or more DOS2 partitions.)

GLOSSARY

Following is a list of terms, used in this manual, that may 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 required command variable that defines or directs that 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 <i>.exe</i> in MS-DOS and you can execute them by typing the program name and pressing <code>ENTER</code> .
Assembler	A program to create binary files from assembly language source code (see assemble).
background	A multitask operation such as printing files (in the background) while entering data.
batch files	A method of combining several commands into 1 file that, when called, will automatically execute 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.

BIOS	Basic Input/Output System. A low level interface between application programs and MS-DOS and your computer hardware (the keyboard, video screen, printer, disks, etc.). BIOS services are available only through machine language programming.
buffer	A temporary storage area, usually in the computer's memory or on disk. Your computer has buffer areas 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 the <code>[ENTER]</code> key.
CON	A device name in MS-DOS referring to the console (screen or keyboard).
current directory	The directory in which you are currently operating. Check Commands (Section II) for ways to change the current directory.
current drive	The operating drive (the drive that contains the disk you are using). Check the Commands (Section II) for ways to change the current drive.
cursor	The screen prompt symbol. In MS-DOS it is a flashing underscore symbol.
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 separates commands and/or parameters from one another. 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 may have several directories, each of which may 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 could 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 VI for a description of error messages.
ext or extension	An appendage to a filename, 1-3 characters long, that provides further identification. Extensions often indicate a type of file such as <i>maillist.bas</i> where <i>.bas</i> is the extension and indicates a BASIC file.
external commands	Procedures or utilities not resident in memory. External commands must be loaded from disk into the computer before they can be executed.
filename	The name given to a file. It may be 1-8 characters long.
filespec	A filename plus its extension, such as <i>MEMOS.bas</i> .
filter	A function that performs a specific operation on data (such as sorting or searching) as it flows between 2 devices (such as disk and a printer), or between files.

Glossary

format	The process of dividing a disk into tracks and sectors and writing on it the system information used to 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. An internal command can be executed 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 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.
pipng	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 %, that will be replaced by user input when a 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). By context, it can also refer to your computer system, including monitor, CPU, drives, and peripheral equipment.
system prompt	Display on the video screen (A>, B>, C>, or D>) that indicates which drive is the current drive and that MS-DOS is waiting for a command.
switch	A command parameter that activates a special function. The command DIR /W will cause 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, or which characters are legal delimiters.
template	A storage area that contains the last MS-DOS command used. The command can thus be called up for editing and reexecution.
toggle	Switch a function on or off. Pressing <input type="button" value="PRINT"/> once sends screen input to the printer. Pressing <input type="button" value="PRINT"/> again turns off this 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 make sure that it has been copied or stored correctly.
wild card	One or more symbols that are used to represent 1 or more variable characters. The wild card ? can be used in a filename to <i>stand in place of</i> a legal filename character.

INDEX

- /CR 107
- /LF 107
- 3 1/2-inch drive, configuring in system 325-327

- active partition 94, 129
- adding lines to a file in memory 209
- alignment 249
- ampersand, used to extend lines with library manager 261
- APPEND 37-38
- appending to library modules 260
- application program 345
- archive attribute, displaying and setting 40
- argument 345
- ASCII 345
- ASCII and scan codes 311-320
- ASCII file 53, 56, 59
- assemble 345
- assembler 236, 345
- ASSIGN 37, 39, 103
- asterisk, used to extract library modules 261
- ATTRIB 40
- AUTOEXEC.BAT 24, 164, 169
- AUX 9, 64

- background 345
- background printing 143-147
- BACKUP 41-45
- backup file, created with EDLIN 197
- backup log entry 43
- BACKUP/RESTORE compatibility 43
- backups, where BACKUP places files 44
- BASIC 252
- batch file 345
 - calling one from another 27
 - conditional execution of commands 114-116
 - creating 21-22
 - displaying message 140-142
 - executing 22
 - including remarks 22, 153
 - pausing 22
 - percent sign 28
 - replaceable parameters 24-27

- suspending execution 140-142
- transferring control to another line 105-106
- where to create 23
- batch file process, summary 23
- baud rates 133
- binary 345
- binary file 53, 56, 59
 - displaying 182
- BIOS 346
- black and white printer 108
- block device, defining parameters for 325-327
- BREAK 46
- buffer 346
- BUFFERS, CONFIG.SYS command 323-324
- byte alignment 249

- canonical frame 249
- canonical frame address 253
- carriage return 107
- CGP-220 printer 107
- character scan codes, converting to US scan codes 118
- characters-per-line, setting 132
- CHDIR 47-48
- CHKDSK 49-50
- class 238-239
- CLS 51
- COBOL 252
- color TV 134
- COM1, COM2 9, 64
- combine types 249
- command 346
 - editing 189-193
 - entering 1
 - entering more than one at a time *see* batch files and command piping
 - executing for several items in a set 97-98
 - maximum number of characters 1
- COMMAND 52
- command error messages 297-305
- command path, setting or displaying 138-139
- command piping 20
- command processor 52
 - specifying a string to execute 52
 - telling not to exit to higher level 52
- command types 29

COMMAND.COM 51
 compiler 236
CON 9, 64, 346
CONFIG.SYS 124, 157, 164, 175, 177
 changing 330
 creating 321
 location 321
 sample file 329
CONFIG.SYS commands
 BREAK 323
 BUFFERS 323-324
 COUNTRY 324
 DEVICE 324-325
 DRIVPARM 325-327
 FCBS 327
 FILES 327-328
 LASTDRIVE 328
 SHELL 328-329
console 9, 64
continuous timeout retries on RS232 port 133
control characters, inserting into text with EDLIN 206
control-c check 46
control keys 3
control, returning from MS-DOS to previous level 81
COPY 13, 53-55, 73
COPY/APPEND 56-58
COPY/COMBINE 59-61
COPYDOS 62-63
copying lines with EDLIN 210-212
COUNTRY 68
country code, selecting 163-164
CPU speed, changing 133, 134, 135
CTTY 64
current directory 14-16, 17, 346
 changing 16, 47-48
 changing to a directory on another disk 48
 displaying 47-48
current drive 14, 346
cursor 346
cylinders 113

databits 133
data file path, displaying 37
data files, setting search path 37-38
data sorting 173-174

DATE 65-66

date

- changing 65-66
- changing format 66
- displaying 65-66
- format for entering 65
- format in which date is stored 65

DEBUG command parameters 265-267

DEBUG commands

- ASSEMBLE 268-270
- COMPARE 270-271
- DUMP 271-272
- ENTER 272-274
- FILL 274-275
- GO 275-276
- HEX 277
- INPUT 277
- LOAD 278-279
- MOVE 279-280
- NAME 280-281
- OUTPUT 282
- PROCEED 282-283
- QUIT 283
- REGISTER 283-286
- SEARCH 286-287
- TRACE 287-288
- UNASSEMBLE 288-289
- WRITE 290-291

DEBUG command summary 265

DEBUG error messages 310

DEBUG, starting 263

default 33, 346

DEL *see* ERASE

deleting

- from the template 193
- library modules 260
- lines with EDLIN 213-215
- strings with EDLIN 228-229

delimiter 34, 346

destination 34

device 346

device drivers *see* installable device drivers

device error messages 293-296

device names 9

DIR 13, 68-69

- directories 5, 347
 - backing up 43
 - changing 47-48
 - checking 49-50
 - checking, sending output to a file 49-50
 - copying 186-188
 - creating 12-13, 127-128
 - current *see* current directory
 - deleting 13, 161-162
 - displaying 10-12, 68-69
 - parent *see* parent directory 17
 - root *see* root directory 6
 - wide display 68-69
- directory names, anonymous 17-18
- disk buffers, setting number 323-324
- disk file 347
- disk, bootable 99, 102, 109
- DISKCOMP 39, 70-71
- DISKCOPY 39, 72-73
- diskettes
 - comparing 70-71
 - copying 72-73
 - formatting 99-104
- DISKTYPE 74
- display 347
- displaying a command line 191-192
- displaying lines with EDLIN 222-224
- DMP 132
- DMP-110 printer 107
- DOS partition *see* FDISK
- DOS2 partition
 - accessing *see* CONFIG.SYS commands, MLPART.SYS
 - creating 130-131
 - formatting 129
- double-sided diskette, formatting as single-sided 99, 102
- drive 347
 - current *see* current drive
 - joining to a pathname 117
 - size and capacity 74
 - support of additional drives 332-333
 - type 74
- drive letter, reassigning 39
- dummy argument 347
- dump 347
- DWP 132

- ECHO 75-76
- editing a file in parts 234
- editing functions
 - EDLIN 2
 - MS-DOS 2
- editing keys 1-2
- EDLIN command format and rules 205-206
- EDLIN command parameters 207-208
- EDLIN commands
 - Append Lines 209
 - Copy Lines 210-212
 - Delete Lines 213-215
 - Edit Line 216-217
 - End Edit 218
 - Insert 219-221
 - List 222-224
 - Move Lines 225
 - Page 226
 - Quit 227
 - Replace String 228-229
 - Search Text 230-232
 - Transfer Lines 233
 - Write Lines 234
- EDLIN command summary 207
- EDLIN editing keys
 - copy all 199
 - copy character 198
 - copy to character 198
 - delete character 199
 - delete to character 200
 - enter line 202-203
 - replace template 201-202
 - void line 200
- EDLIN error messages 305-307
- empty subdirectories, copying 186-188
- end-of-line character 107
- environment size 51
- environment variable settings, displaying 165
- environment variable, setting equal to a string 165-168
- EOF character 53, 56, 59
- ERASE 13, 77-78
- error 347
- ERRORLEVEL 114
- error messages 293-310
 - commands 297-305

- DEBUG 310
- EDLIN 305-307
 - linker 308-310
- errors that CHKDSK corrects 50
- Escape Code Sequences 124-125
- EXE2BIN 79-80
- executable file, converting to binary file format 79-80
- EXIST 114
- EXIT 81
- extended printer capabilities 334-338
- extended screen and keyboard features 332
- extending lines in library manager command 261
- extension 6-7, 347
- external commands 29, 347
 - order of search 138
 - transferring to another disk 54
- extracting library modules 261

- FC 82-92
- FDISK 93-94, 111, 112, 343
- File Allocation Table 151
- file control blocks 327-328
- file editing 195-203
- file handles 327-328
- filename extension *see* extension
- filenames 6-7, 347
- files
 - adding to backup 43-44
 - appending to existing files 56-58
 - checking 49
 - combining 59-61
 - comparing 82-92
 - copying 53-55, 186-188
 - creating with EDLIN 9-12, 195-196
 - deleting 77-78
 - displaying contents 10-11, 181, 182
 - displaying date and time modified 68
 - editing with EDLIN 197
 - inserting text with EDLIN 196
 - making minor changes to files 137
 - merging with EDLIN 233
 - packing on backup diskette 43-44
 - paging through with EDLIN 226
 - patching 137
 - recovering 151-152

- renaming 154
- restoring backed up files 157-160
- saving with EDLIN 196
- updating 155-156
- verifying 184
- file sharing 169
 - allowing over a network 40
- filespec 347
- file system 5
- filter 20, 347
- FIND 20, 95-96
- floppy diskette *see* diskette
- FOR 97-98
- FORMAT 112, 343
- format 348
- FORMAT 73, 93, 99-104
- FORTRAN 242, 243, 252
- function keys 189-190
 - copy all 189
 - copy character 189
 - copy to character 189
 - delete character 189
 - delete to character 189
 - end-of-file 190
 - enter line 190
 - insert 189
 - replace template 190
 - void line 190
- global symbols 243
- GOTO 105-106
- GRAPHICS 107-108
- groups 238-239
- hard disk backup 41-45
- hard disk formatting 102 *see also* FORMAT, hard disk setup, HFORMAT, and MLFORMAT
 - formatting track and sector information *see* HSECT
 - locking out flawed areas 109, 113
 - using entire disk for MS-DOS 94
 - using multiple operating systems 93
- hard disk heads 113
 - parking 172

-
- hard disk partitions
 - accessing DOS2 partitions *see* CONFIG.SYS commands, MLPART.SYS
 - creating *see* FDISK, hard disk setup, and MLPART
 - formatting *see* FORMAT, hard disk setup, HFORMAT, and MLFORMAT
 - maximum number 93
 - status 94, 129
 - hard disk setup 343-344
 - HFORMAT 93, 109-110, 111, 343
 - home directories 16-17
 - changing 47-48
 - HSECT 93, 111-113, 343

 - IF 114-116
 - input 348
 - input/output device, changing 64
 - input/output, redirecting 19
 - inserting text
 - in a command line 192
 - with EDLIN 219-221
 - installable device drivers
 - ANSI.SYS 332
 - configuring in system 324-325
 - DRIVER.SYS 332-333
 - KEYCNVRT.SYS 333-334
 - LPDRVR.SYS 334-338
 - MLPART.SYS 338-339
 - SPOOLER.SYS 339
 - summary of drivers 331
 - VDISK.SYS 340-341
 - internal 348
 - internal commands 29
 - internationally configured MS-DOS diskette 163-164

 - JOIN 39, 103, 117

 - key conversion 333-334
 - keyboard device 9
 - keyboard layout code, selecting 163-164
 - keyboard program, replacing with an international program 118 -119
 - keys
 - backspace 3
 - control-c 3, 34, 70, 242, 261, 264
-

- control-h 3
- control-j 3
- control-n 3
- control-p 3
- control-s 3, 264
- escape 3
- hold 3, 34
- print 3
- shift-print 3
- KEYTXX 118-119
- keywords 35
 - synonymous 36
- LABEL 39, 120
- LASTDRIVE 177
- less-than sign 19
- LF 121
- libraries 241
- library manager 256-261
 - command characters 260-261
 - order of operations 256
 - starting with a command line 258-259
 - starting with a response file 259-260
 - starting with the prompt method 257-258
- line feed 107
 - suppressing after carriage return 121
- line printer 9
- linefeed, setting for printer 132
- lines, specifying in EDLIN commands 208
- linker
 - alignment 249
 - assigning addresses 253
 - command characters 241-242
 - command prompts 239-241
 - error messages 308-310
 - loading segments 251-253
 - relocation fixups 254-255
 - sample session 246-248
 - starting with a command line 244-245
 - starting with a response file 245-246
 - starting with prompts 244
 - types of references 254-255
- linker combine types
 - common 251
 - private 250

- public 250
- stack 250
- linker switches
 - /DSALLOCATE 242
 - /HIGH 243
 - /LINENUMBERS 243
 - /MAP 243
 - /NO 244
 - /PAUSE 243
 - /STACK 243-244
- list file 236, 241
- loading a line for editing with EDLIN 216-217
- long references 255
- LPDRVR.SYS 123, 124
- LPSETUP 122-126
 - Escape Code Sequences 124-125
- LPT 9, 348

- machine language code 235
- MCOPY.EXE 187
- Media Error Map 113
- media types, backing up between different types 43-45
- memory combine type 253
- merging files with EDLIN 233
- messages, displaying during batch file execution 75-76
- minus sign, used to delete library modules 260
- MKDIR 12-13, 127-128
- MLFORMAT 129, 344
- MLPART (MLPART.COM) 130-131, 344
- MLPART.SYS 129, 130, 344
- MODE 132-135
- modified files
 - backing up 41-43, 45
 - copying 186-188
- monochrome monitor, using color-oriented software 134, 135
- MORE 20, 136
- moving lines with EDLIN 225
- MS-DOS commands, quick reference 30-32
- MS-DOS version, displaying 182
- MS-LIB *see* library manager 256-261
- multiple DOS2 partitions, accessing 338-339

- near segment-relative references 255
- near self-relative references 254
- networking 169

- NL 132
- non-active partition 94, 129
- notation 35-36
- NUL 9

- object modules 240
- operating system files *see* system files
- output 348
- overtyping in a command line 192

- pagination 122-126
- paging through a file with EDLIN 226
- paragraph alignment 249
- parameter 348
- parameters 33
- parent directory 17
- parity 133
- partitions *see* hard disk partitions
- Pascal 242, 243, 244, 252
- PATCH 137
- PATH 138-139
- path 348
- pathname 8-9, 348
 - substituting a virtual drive name for a pathname 177
- PAUSE 22, 140-142
- period
 - to indicate current directory 17
 - to indicate current line 208
- periods, to indicate parent directory 17
- peripheral 348
- pipng 348
- plus sign
 - used to append to library modules 260
 - used to extend lines in linker command 241
- pound sign, used to indicate last line of program 208
- PRINT 39, 143-147
- print queue 143-147
- print spooler 175-176
- printer
 - characters-per-line 133, 135
 - compatibility 107-108
 - control codes 335-337
 - driver 334-338
 - linefeed 132, 134
 - parameters 134

- timeout delay 133, 135
- type 107, 132
- printer data, buffering 339
- printer output, redirecting to serial port 135
- PRN 9, 348
- program 348
 - executing 1
- PROMPT 148-150
- public symbols 243

- quitting EDLIN 227

- read-only attribute 40
- RECOVER 151-152
- relative line numbers in EDLIN 205
- REM 22, 153
- RENAME 154
- REPLACE 155-156
- replaceable parameters 348
 - using more than ten 170-171
- replacing strings with EDLIN 228-229
- RESTORE 157-160
 - finding which disk to restore 43
- RMDIR 13, 161-162
- root directory 6, 349
 - joining to a different pathname 117
- RS-232 ports 9, 64
- RS232 communication parameters, setting 133, 135
- run file 240

- saving a file after editing with EDLIN 218
- saving in the template 192
- screen scroll 34
- screen
 - clearing 52
 - displaying 20, 136
 - shifting left and right 132
- searching for strings with EDLIN 230-232
- sector 151-152
- sectors, verifying after copy 54, 57, 59
- segment addresses 253
- segments 238
 - how the linker loads segments 251-253
- SELECT 163-164

semicolon
 in linker 242
 in library manager 261
serial port 9
SET 165-168
SHARE 169
SHIFT 25, 170-171
SHIPTRAK 172
short references 254
SORT 20, 173-174
source 34
source code 235
special keys 1-2
special type 35-36
SPOOLER.SYS 175-176
stopbits 133
strings 208, 349
 deleting with EDLIN 228-229
 replacing with EDLIN 228-229
 searching for with EDLIN 230-232
subdirectories
 copying 186-188
 displaying 181
 restoring 157, 159
SUBST 39, 103, 177
switches 33, 349
syntax 35, 349
SYS 178
system 349
system disk, creating 178
system files, copying 62
system prompt 349
 changing 148-150

target 34
template 191, 349
temporary file 237
text file *see* ASCII file
text
 finding 20, 95-96
 sorting 20
TIME 179-180
toggle 349
track 349
TREE 181

TYPE 182

US scan codes 164

VER 183

VERIFY 184

verify 349

verifying sectors after copy 54, 57, 59

video display, shifting left and right 134

video mode, setting 132, 134

video palette color, changing 132, 135

video scan lines, setting 132, 134

virtual disks 340-341

virtual drive name 177

VM.TMP 237

VOL 185

volume label 68, 99, 102, 109, 126-127, 185

wild card 8, 154, 349

word alignment 249

writing edited lines to disk 234

XCOPY 73, 186-188

