

MINDSET

MSTM-DOS
Reference Manual

Personal
Computer
System

100201-001

Information in this document is subject to change without notice and does not represent a commitment on the part of Mindset Corporation. The software described in this document is furnished under a license agreement or non-disclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the MS-DOS Disk Operating System on cassette tape, disk, or any other medium for any purpose other than the purchaser's personal use on the Mindset Personal Computer System.

LIMITED WARRANTY

Mindset Corporation shall have no liability or responsibility to purchaser or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by this product, 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 this product. This product will be exchanged within 90 days from date of purchase if defective in manufacture, labeling or packaging, but except for such replacement the sale or subsequent use of this program is without warranty or liability.

THE ABOVE IS A LIMITED WARRANTY AND THE ONLY WARRANTY MADE BY MINDSET CORPORATION. ANY AND ALL WARRANTIES FOR MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY EXCLUDED.

MINDSET is a trademark of Mindset Corporation.

Copyright © 1983, Mindset Corporation.

All rights reserved.

Printed in U.S.A.

100201-001

A Tec-Ed Manual.

Contents

Section 1 **Introduction**

What Is MS-DOS?	1-1
What Is an Operating System?	1-1
How to Use this Manual	1-2
Documentation Conventions	1-3
MS-DOS Files	1-4

Section 2 **Getting Started**

Loading MS-DOS	2-1
Entering the Date and Time	2-2
Changing the Default Drive	2-4
Formatting Your Disks	2-4
The FORMAT Command	2-5
Assigning Volume Labels	2-6
Backing Up Your Disks with DISKCOPY	2-7
Automatic Program Execution	2-9
Files	2-9
How MS-DOS Keeps Track of Your Files	2-9
The DIR (Show Directory) Command	2-10
The CHKDSK (Check Disk) Command	2-11
Turning Off the System	2-11
Summary of Commands in this Section	2-12

Section 3**More About Files**

Naming Your Files	3-1
Wild Cards	3-3
The ? Wild Card	3-3
The * Wild Card	3-3
Illegal Filenames	3-4
Copying Your Files	3-5
Protecting Your Files	3-6
Directories	3-6
Filenames and Paths	3-9
Pathnames	3-9
Pathing and External Commands	3-10
Pathing and Internal Commands	3-11
Displaying Your Working Directory	3-12
Creating a Directory	3-13
Changing Your Working Directory	3-13
Removing a Directory	3-13
Summary of Commands in this Section	3-14

Section 4**Learning About Commands**

Introduction	4-1
Types of MS-DOS Commands	4-2
Internal Commands	4-2
External Commands	4-2
Command Options	4-3
Information Common to All MS-DOS Commands	4-4
Batch Processing	4-5
The AUTOEXEC.BAT File	4-7
Creating an AUTOEXEC.BAT File	4-8
Creating a .BAT File with Replaceable Parameters	4-9
Executing a .BAT File	4-10
Input and Output	4-11
Redirecting Your Output	4-11
Filters	4-12
Command Piping	4-13
Summary of Commands in this Section	4-14

Section 5**MS-DOS Commands**

Documentation Conventions for Commands	5-1
MS-DOS Command Summary	5-2
Batch Commands (Command Extensions) Summary	5-4
MS-DOS Command Descriptions	5-5
Batch Processing Commands	5-51

Section 6**MS-DOS Editing and Function Keys**

Special MS-DOS Editing Keys	6-1
Control Sequences	6-5

Section 7**The Line Editor (EDLIN)**

Introduction	7-1
Starting EDLIN	7-2
Special Editing Keys	7-3
Command Information	7-14
Command Options	7-16
EDLIN Commands	7-17
Error Messages	7-44

Section 8**File Comparison Utility**

Introduction	8-1
Limitations on Source Comparisons	8-1
File Specifications	8-2
Using FC	8-2
FC Switches	8-3
Difference Reporting	8-5
Redirecting FC Output to a File	8-5
Example 1	8-6
Example 2	8-8
Example 3	8-9
Error Messages	8-10

Section 9

The Linker Program (MS-LINK)

Introduction	9-1
Overview of MS-LINK	9-1
Definitions You'll Need to Know	9-2
Files that MS-LINK Uses	9-5
Input File Extensions	9-6
Output File Extensions	9-6
VM.TMP (Temporary) File	9-6
Starting MS-LINK	9-7
Summary of Methods to Start MS-LINK	9-7
Method 1: Prompts	9-7
Method 2: Command Line	9-8
Method 3: Response File	9-9
Command Characters	9-10
Plus Sign	9-10
Semicolon	9-11
CTRL-C	9-11
Command Prompts	9-12
MS-LINK Switches	9-14
Sample MS-LINK Session	9-16
Error Messages	9-18

Section 10

Library Manager

Features of MS-LIB	10-1
Overview of MS-LIB Operation	10-2
Running MS-LIB	10-4
Starting MS-LIB	10-5
Summary of Methods to Start MS-LIB	10-5
Method 1: Prompts	10-5
Method 2: Command Line	10-6
Method 3: Response File	10-7
Command Prompts	10-8
Command Characters	10-10
Plus Sign	10-10
Minus Sign	10-11
Asterisk	10-11
Semicolon	10-11
Ampersand	10-12
CTRL-C	10-12
Error Messages	10-13

Appendix A
Instructions for Users with Single-Drive Systems

Appendix B
Disk Errors

Appendix C
ANSI Escape Sequences

Cursor Functions	C-2
Erasing	C-4
Modes of Operation	C-4
Keyboard Reassignment	C-6

Appendix D
How to Configure Your System

Index



Figures and Tables

Figure 1-1:	Hardware/Software relationships	1-2
<hr/>		
Figure 2-1:	The DISKCOPY command	2-8
<hr/>		
Figure 3-1:	Copying files to another disk	3-5
Figure 3-2:	Copying files on the same disk	3-6
Figure 3-3:	A sample hierarchical directory structure	3-8
<hr/>		
Figure 4-1:	MS-DOS batch file steps	4-6
Figure 4-2:	How MS-DOS uses the AUTOEXEC.BAT file	4-8
<hr/>		
Figure 6-1:	Command line and template	6-2
Table 6-1:	Special Editing Functions	6-2
Table 6-2:	Control Sequences	6-5
<hr/>		
Table 7-1:	Special Editing Keys	7-3
Table 7-2:	EDLIN Commands	7-15
<hr/>		
Figure 9-1:	The MS-LINK operation	9-3
Figure 9-2:	How memory is divided	9-4
<hr/>		
Figure 10-1:	MS-LIB operation	10-4



Introduction

Getting Started

More About Files

Section 1

Introduction

What Is MS-DOS?

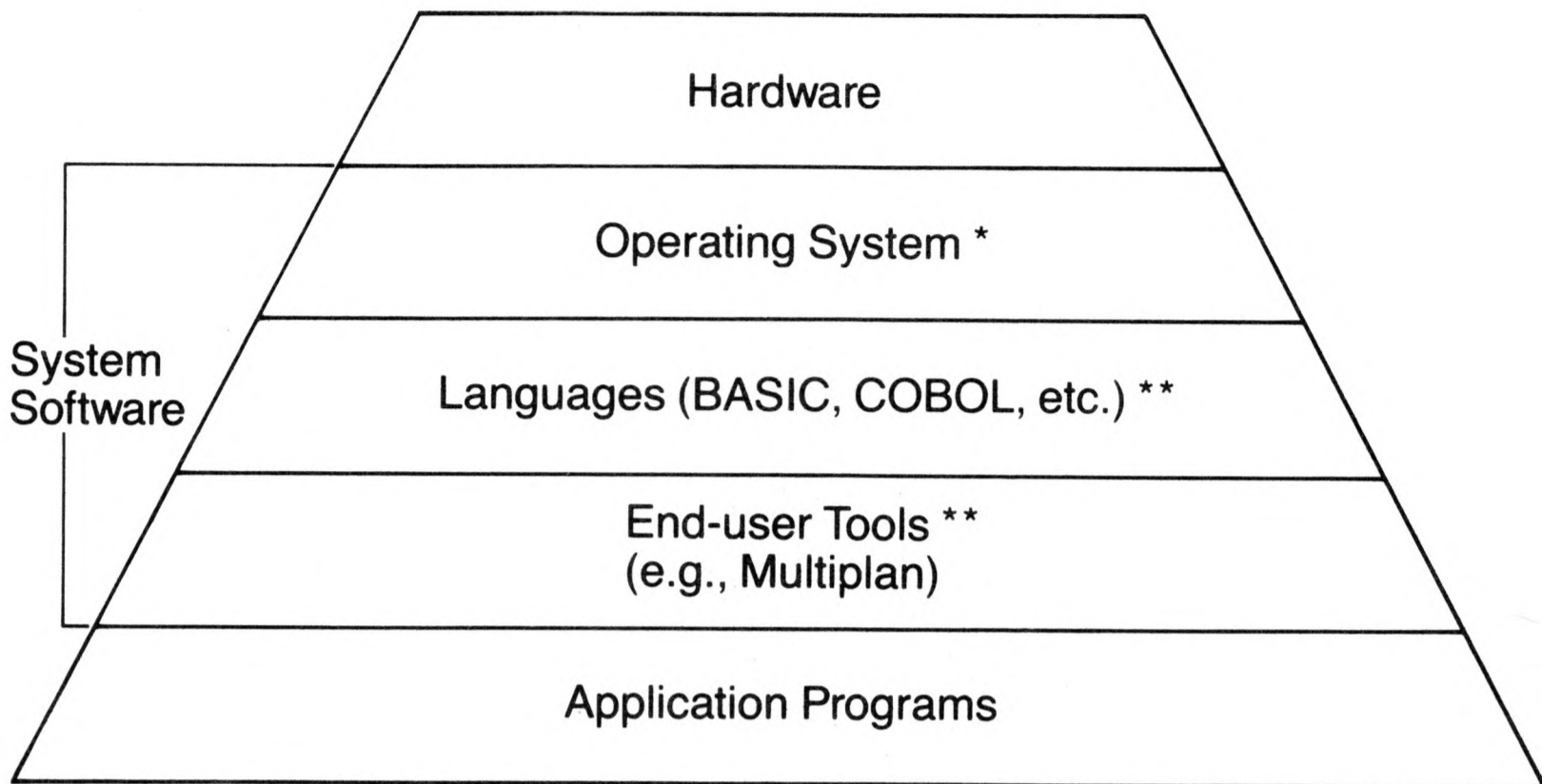
MS™-DOS is a disk operating system (DOS) for your Mindset Personal Computer System. Through MS-DOS, you communicate with the computer, disk drives, and printer, managing these resources to your advantage.

What Is an Operating System?

An operating system is your “silent partner” when you are using the computer. It provides the interface between the hardware and both you (the user) and the other system software.

An operating system is the piece of system software most closely associated with the hardware. The Mindset OS is unique to the Mindset microprocessor, the Intel 80186. Figure 1-1 illustrates how the hardware, the system software, and the application software are related.

MS-DOS is a disk operating system that enables you to create and keep track of files, run and link programs, and access peripheral devices (for example, printers and disk drives) that are attached to your computer.



* Must adapt to new hardware

** If adapted to operating system, these don't change

Figure 1-1: Hardware/Software relationships

How to Use this Manual

This manual describes MS-DOS and how to use it. This section introduces some basic MS-DOS concepts; Section 2 discusses how to start using MS-DOS and how to format and back up your disks.

Section 3 tells you about files—what they are and how to use them. Sections 4 through 6 introduce MS-DOS commands, and Section 7 describes the line editor, EDLIN. Read these sections carefully—they contain information on protecting your data, system commands, and the MS-DOS editing commands.

Section 8 explains how to use the MS-DOS File Comparison Utility, FC. This utility is helpful when you need to compare the contents of two source or binary files.

If you want to write programs and link separately produced object modules and create relocatable modules, read Section 9 for a description of a useful MS-DOS utility, MS-LINK. Section 10 describes a library manager, MS-LIB.

Appendices to this manual include instructions for users with one-disk-drive systems and descriptions of disk error messages.

Documentation Conventions

The following conventions are used throughout this manual in descriptions of command and statement syntax:

- [] Square brackets indicate that the enclosed entry is optional.
- < > You supply the text for any items enclosed in angle brackets. For example, you should enter the name of your file when <filename> is shown in the format.
- When text is enclosed in a box, you must press the key named by the text. For example, RETURN .
- BOLD** Bold lettering indicates text you must enter.
- { } Braces indicate that you have a choice between two or more entries. At least one of the entries enclosed in braces must be chosen unless the entries are also enclosed in square brackets.
- ... Ellipses indicate that an entry may be repeated as many times as needed or desired.
- | A bar indicates an OR statement in a command. When used with an MS-DOS filter, the bar indicates a pipe.
- CAPS Capital letters indicate portions of statements or commands that must be entered exactly as shown.

All other punctuation, such as commas, colons, slash marks, and equal signs, must be entered exactly as shown.

MS-DOS Files

The MS-DOS disk contains the following files:

File Name	Function of File
COMMAND.COM	MS-DOS command processor
* MSDOS.SYS	MS-DOS operating system
* IO.SYS	Hardware to operating system interface
EDLIN.COM	Line editor
DEBUG.COM	Debugger
LINK.EXE	Linker
CHKDSK.COM	Checks disks
FORMAT.COM	Formats disks
SYS.COM	Transfers system
DISKCOPY.COM	Backup utility
RECOVER.COM	Recovers disks
PRINT.COM	Print spooler
MORE.COM	Reviews text
SORT.EXE	Sorts text
FIND.EXE	Finds a string in a list of files or standard input
EXE2BIN.EXE	Converts .EXE files
CONFIG.SYS	System configuration file
FC.EXE	Compares files
MODE.COM	Sets screen, printer modes
MYFILE1.TXT	Sample text file

*These files are "hidden", which means that they do not appear in the MS-DOS directory even though they are present on the disk.

You will recognize this list of files when you have learned the DIR (Show Directory) command described in the next section.

In the next section, you learn how to start your MS-DOS system and how to format and back up your disks.

Getting Started

Loading MS-DOS

To begin loading MS-DOS, insert your MS-DOS diskette into disk drive A (the left drive on a two-drive system). Turn on your Mindset Personal Computer or press the ALT-RESET keys if it is already turned on. If you selected the disk as the highest load priority, then the Mindset Personal Computer loads MS-DOS immediately. If you selected one or both cartridge slots to have higher load priorities than the disk, then the Mindset Personal Computer checks each cartridge slot for a program before it loads MS-DOS from the disk (see Section 4, "System Configuration", in the System Unit Operation Guide).

Loading MS-DOS takes about 5 seconds. Once MS-DOS is loaded, the system searches the MS-DOS disk for the COMMAND.COM file and loads it into memory. The COMMAND.COM file is a program that processes the commands you enter and then runs the appropriate programs. It is also called the command processor.

When the command processor loading is complete, you see the following display on your screen (the underscore represents the cursor):


```
MS-DOS Version 2.xx  
Copyright 1981, 1982 Microsoft Corp.  
Copyright 1983, Mindset Corp.
```

```
Command V. 2.02  
Current date is Mon 1-02-1984  
Enter new date: __
```

You can update the date and time at your keyboard.

Entering the Date and Time

Type today's date in a mm-dd-yy format, where:

mm is a one- or two-digit number from 1 to 12 (representing the month).

dd is a one- or two-digit number from 1 to 31 (representing the day of month).

yy is a two-digit number from 80 to 99 (the 19 is assumed), or a four-digit number from 1980 to 2099 (representing the year).

Any date is acceptable in answer to the "Enter new date:" prompt as long as it follows the preceding format. You can use hyphens (-) or slashes (/) as separators between the numbers. For example:

6-1-82 and **06/01/82**

are both acceptable answers to the "Enter new date:" prompt.

If you enter an invalid date or form of date, the system prompts you again with "Enter new date:".

After you respond to the "Enter new date:" prompt and enter your answer by pressing the RETURN key, you see a prompt similar to this:

```
Current time is 8:30:14.32
Enter new time: __
```

This time display indicates that the current time is 14 and $\frac{32}{100}$ seconds past 8:30 a.m.

Enter the current time in the hh:mm:ss format, where:

hh is a one- or two-digit number from 0 to 23 (representing hours).

mm is a one- or two-digit number from 0 to 59 (representing minutes).

ss is a one- or two-digit number from 0 to 59 (representing seconds).

You do not need to enter the seconds or hundredths of seconds. The system sets them to 00.00 if you do not enter them before you press the RETURN key to enter the time values.

MS-DOS uses this time value to keep track of when you last updated or created files on the system. Notice that MS-DOS uses military time; for example, you would enter 1:30 p.m. as 13:30. Enter the following time:

```
Current time is 0:00:14.32
Enter new time: 9:05 RETURN
```

Use only a colon to separate hours and minutes. If you enter an invalid number separator, MS-DOS repeats the prompt.

Note: If you make a mistake while typing, hold down the control key (CTRL) on your keyboard and press the C key. This CTRL-C sequence aborts your current entry. You can then re-answer the prompt or type another command. To correct a line before you press RETURN, use the BACK SPACE key to erase one letter at a time.

Note: If you do not want to enter date information, you can simply press the RETURN key and skip over the "Enter new date:" prompt.

You have now completed the steps for starting MS-DOS.

Changing the Default Drive

After you have answered the "Enter new time:" prompt, a message appears that looks like this:

```
A> _
```

The A> is the MS-DOS prompt from the command processor. This prompt tells you that MS-DOS is ready to accept commands.

The A in the previous prompt represents the default disk drive. It indicates that MS-DOS searches only the disk in drive A for any filenames you may enter and writes files to that disk unless you specify a different drive.

You can ask MS-DOS to search the disk in drive B by changing the default drive designation or by specifying B: in a command. To change the default disk drive designation, enter the new drive letter followed by a colon. For example:

```
A> (MS-DOS prompt)
A>B:  (You have typed B: in response to the prompt)
B> (The system responds with B and drive B is now the
    default drive)
```

The system prompt B> appears and MS-DOS searches only the disk in drive B until you specify a different default drive.

If only one disk drive is attached to your computer, turn to Appendix A, "Instructions for Users with Single-Drive Systems", for important information.

Formatting Your Disks

You must "format" all new disks before they can be used by MS-DOS.

A blank disk must be formatted with the MS-DOS FORMAT command. The FORMAT command sets up the disk so that MS-DOS can use it to store and retrieve information; it also analyzes the disk for defective tracks. If the disk is not already blank, formatting it destroys any data that exists on the disk. Although formatting can be a convenient way to make

a disk blank, it is recommended that the MS-DOS DELETE command be used for this purpose. Refer to Section 5, "MS-DOS Commands", for more information on the DELETE command.

The FORMAT Command

The syntax of the FORMAT command is:

```
FORMAT [d:] [/V] [ { /8 } ] [/1][/0][/C][/D][/S]
```

where:

- d: is the drive designation (the drive that contains the disk to be formatted).
- /V is a switch which indicates that you want to specify a volume label as part of the formatting procedure.
- /8 or /9 is a switch which indicates whether to format the disk with 8 or 9 sectors per track, respectively. /9 is the default value.
- /1 is a switch which forces the formatting to occur on only one side of the diskette, even though the drive or diskette is double-sided. This switch is especially useful for compatibility with single-sided disk drive systems.
- /0 is a switch which causes the disk directory to be initialized similarly to earlier versions of MS-DOS. Used only for compatibility, this switch causes performance degradation on the Mindset computer.
- /C is a switch which indicates that formatting is not required. You need only to re-initialize the directories to create a "clean" disk. The /C option cannot be used with the /8, /9, or /1 options.
- /D is a switch which causes the file CONFIG.SYS to be copied to the disk in Drive B.
- /S is a switch which indicates that you want the system to copy the "hidden" MS-DOS system files from the disk in the default drive to the newly formatted disk.

Note that the brackets identify optional information. If you do not specify a disk drive (for example, A: or B:), MS-DOS formats the disk that is in the default drive.

With the MS-DOS disk already in drive A, you are ready to format your new blank disk. The following command formats the new disk in drive B.

FORMAT B:

MS-DOS issues the following message:

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

After you insert the new disk in drive B and press any key on the keyboard, the system responds:

Formatting...

while MS-DOS is formatting your disk.

Assigning Volume Labels

If you add the /V switch to the FORMAT command, MS-DOS asks for the volume label of the disk. (See below.) The /S switch tells MS-DOS to copy its system (“hidden”) files onto the new disk.

When the formatting is finished, MS-DOS issues a message similar to this:

Formatting...Format complete System transferred

If you have used the /V switch, MS-DOS displays:

Volume label (11 characters. RETURN for none)?

Volume labels are useful to identify disks—they are like a name tag for each disk. When you assign a unique volume label to a disk, you can always be sure that you know which disk you are using. You can display the volume label you assign to a disk by issuing the MS-DOS VOL command (refer to Section 5, “MS-DOS Commands”, for more information on the VOL command).

Type a volume label in response to the preceding prompt if you want to use this method to identify the disk, and press RETURN. An example of a volume label is PROGRAMS. If you do not want to use a volume label to

identify this disk, simply press the RETURN key. MS-DOS displays on your screen a message similar to this:

160256 bytes total disk space
12800 bytes used by system
143360 bytes available on disk

Format another (Y/N)?__

Type **Y** to format another disk. If you are finished formatting your disks, type **N** to end the FORMAT program and return to the MS-DOS system prompt.

Note: Do not open the disk drive latch until this message appears.

Backing Up Your Disks with DISKCOPY

It is strongly recommended that you make backup copies of all your disks. If a disk becomes damaged or if files are accidentally erased, you still have all of the information on your backup disk. You should also make a backup copy of your MS-DOS disk. You can back up disks by using the MS-DOS DISKCOPY command.

The DISKCOPY command copies the contents of a disk onto another disk. You can use this command to duplicate both the MS-DOS disk and a disk that contains your own files. DISKCOPY is the fastest way of copying a disk because it copies the entire disk in one operation, including any hidden MS-DOS system files.

The format of the DISKCOPY command is:

DISKCOPY [drive1:] [drive2:]

Drive1 is the disk drive that contains the disk that you want to copy; drive2 is the disk drive that contains the blank or "destination" disk. The blank disk must be formatted prior to running DISKCOPY.

For example, if you want to make a copy of your MS-DOS disk which is in drive A, type:

DISKCOPY A: B:

(Be sure to include all spaces shown.) MS-DOS responds:

Insert source diskette into drive A:
Insert formatted target diskette into drive B:
Press any key when ready

Make sure the MS-DOS disk is in drive A and insert a blank, formatted disk in drive B. Then press any character key. MS-DOS begins copying the MS-DOS disk. After MS-DOS copies the disk, MS-DOS displays:

Copy complete
Copy another (Y/N)?

Type **Y** if you wish to copy another disk with DISKCOPY. If you type **N** the DISKCOPY program ends and MS-DOS displays the system prompt.

You now have a duplicate copy of your MS-DOS disk in drive B. You can save this duplicate copy as your backup copy of the MS-DOS disk.

Figure 2-1 illustrates the operation of the DISKCOPY command:

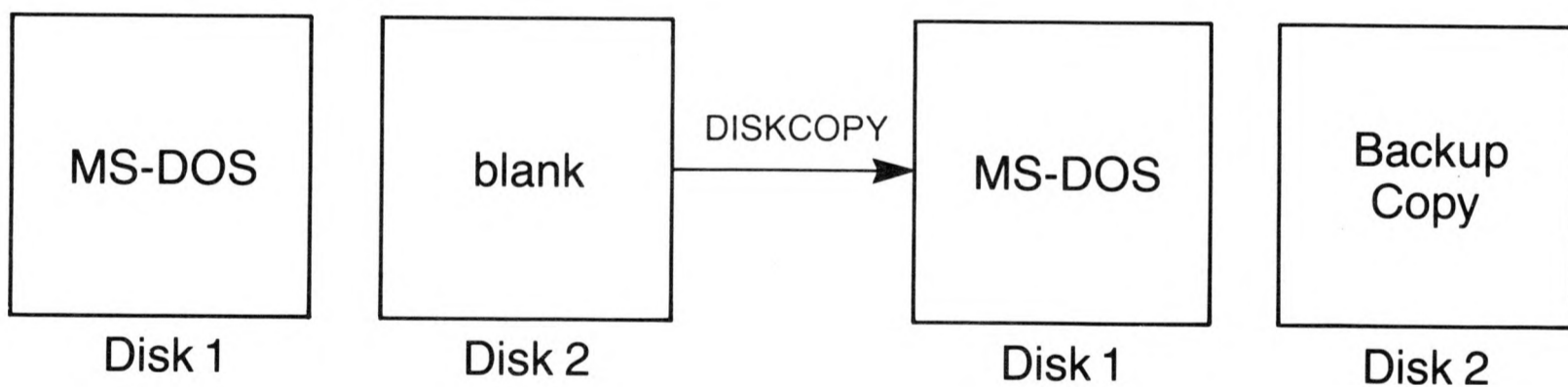


Figure 2-1: The DISKCOPY command

Disks must be the same size and density to be copied with the DISKCOPY command. Refer to Section 5, "MS-DOS Commands", for more information on the DISKCOPY command.

Note: If either of the disks that you are using has defective tracks, DISKCOPY does not work. Use the COPY command to back up your disks in these cases. (COPY skips over defective tracks.) Refer to Section 5, "MS-DOS Commands", for information on how to use COPY to back up your disks.

Automatic Program Execution

To run a specific program automatically each time you start MS-DOS, you can use Automatic Program Execution. For example, you may want MS-DOS to display the names of your files each time you load MS-DOS.

When you start MS-DOS, the command processor searches for a file named AUTOEXEC.BAT on the MS-DOS disk. If this file is present, then MS-DOS runs it first each time MS-DOS is started. Section 4, "Learning About Commands", tells you how to create an AUTOEXEC.BAT file.

Files

A file is a collection of related information. A file on your disk is comparable to a file folder in a desk drawer. For example, one file folder might contain the names and addresses of the employees who work in the office. You might name this file the Employee Master File. A file on your disk could also contain the names and addresses of employees in the office and could be named Employee Master File.

All programs, text, and data on your disk reside in files and each file has a unique name. You refer to files by their names. Section 3, "More About Files", tells you how to name your files.

How MS-DOS Keeps Track of Your Files

MS-DOS keeps the names of files in directories on a disk. These directories also contain information about the size of the files, their location on the disk, and the dates that they were created and updated. The directory you are working in is called your current or working directory.

An additional system area on each disk is called the File Allocation Table. It keeps track of the location of your files on the disk. It also allocates the free space on your disk so that you can create new files.

These two system areas, the directories and the File Allocation Table, enable MS-DOS to recognize and organize the files on your disks.

MS-DOS copies the File Allocation Table onto a new disk when you format it with the MS-DOS FORMAT command. MS-DOS then creates a single empty directory called the root directory.

The DIR (Show Directory) Command

If you want to know what files are on your disk, you can use the DIR command. This command tells MS-DOS to display all the files in the current directory on the disk that is named (not including any hidden system files). For example, if your MS-DOS disk is in drive A and you want to see the listing for the current directory on that disk, type:

DIR RETURN

MS-DOS responds with a directory listing of all the files in the current directory on your MS-DOS disk. The display should look similar to this:

Volume in drive A is DOS 2-0
Directory of A:

COMMAND	COM	16276	10-29-81	11:48a
DEBUG	COM	11534	10-28-82	9:21a
CHKDSK	COM	6272	10-26-82	12:12p
SYS	COM	1400	10-29-82	6:30p
EDLIN	COM	4419	1-01-80	12:41a
RECOVER	COM	2281	10-29-82	5:37p
PRINT	COM	3899	10-27-82	12:19p
LINK	EXE	41856	8-31-82	1:14p
FORMAT	COM	5605	10-28-82	9:55a
EXEFIX	COM	1350	10-06-82	2:57p
SORT	EXE	1280	10-27-82	3:18p
MORE	COM	291	10-27-82	3:20p
FIND	EXE	5888	01-01-80	12:57a
CONFIG	SYS	33	10-18-82	5:02p
LOCATE	EXE	5888	10-27-82	12:53p
FC	EXE	10624	10-27-82	7:00p
LOGIN	COM	299	10-18-82	6:30p

17 File(s) 23040 bytes free

Column 3 indicates the amount of data within the file. The system reserves a minimum of 1 allocation unit for each file. An allocation unit is 512 bytes for a single-sided diskette and 1024 bytes otherwise.

Note: Two MS-DOS system files, IO.SYS and MSDOS.SYS, are “hidden” files that do not appear when you issue the DIR command.

You can also get information about any file on your disk by typing DIR and a filename. For example, if you create a file named MYFILE.TXT on the default drive, the command:

DIR MYFILE.TXT

gives you a display of all the directory information (name of file, size of file, date last edited) for the file MYFILE.TXT.

For more information on the DIR command, refer to Section 5, "MS-DOS Commands".

The CHKDSK (Check Disk) Command

You can use the MS-DOS command CHKDSK to check your disks for consistency and errors, much like a secretary proofreading a letter. CHKDSK analyzes the directories and the File Allocation Table on the disk that you specify. It then produces a status report of any inconsistencies, such as files which have a non-zero size in their directory but really have no data in them.

To check the disk in drive A, type:

CHKDSK A:

MS-DOS displays a status report and any errors that it finds. You can find an example of this display and more information on CHKDSK in the description of the CHKDSK command in Section 5. You should run CHKDSK occasionally for each disk to ensure the integrity of your files.

Turning Off the System

There is no "logoff" command in MS-DOS. When you are finished using your system, end all programs so that the default drive prompt is on the screen. Next, open the disk drive doors and remove the disks. Then, simply turn off your system.

Note: Always remove your disks from the disk drives before you turn off your system.

Summary of Commands in this Section

Command	Purpose	Syntax
FORMAT	Formats disks for MS-DOS	FORMAT [d:][/V] [{ /8 }] [/S][/1][/0][/C][/D] [{ /9 }]
DISKCOPY	Copies disks	DISKCOPY [drive1:] [drive2:]
DIR	Lists directory information	DIR [d:][filename]
CHKDSK	Check for errors on a disk	CHKDSK [d:]

In the next section, you learn more about MS-DOS files.

Section 3

More About Files

In Section 2, you learned that directories contain the names of your files. In this section, you learn how to name and copy your files. You also learn more about the MS-DOS hierarchical directory structure which makes it easy for you to organize and locate your files.

Naming Your Files

The name of a typical MS-DOS file looks like this:

NEWFILE.EXE

The name of a file consists of two parts. The filename is NEWFILE and the filename extension is EXE.

A filename can be from 1 to 8 characters long. The filename extension can be three or fewer characters. A period separates the filename and its extension. You can type any filename in lowercase or uppercase letters; MS-DOS always translates these letters into uppercase characters. The combination of a filename and an extension is called a file specification (or filespec).

You can include a drive designation along with a filespec. A drive designation tells MS-DOS to look on the disk in the designated drive to find the filespec typed. For example, to find directory information about the file NEWFILE.EXE which is located on the disk in drive A (and drive A is not the default drive), type the following command:

DIR A:NEWFILE.EXE RETURN

MS-DOS then displays directory information about the file NEWFILE.EXE.

The drive designation A: is not necessary if A is the default drive (though you can successfully execute the command either way).

Your filenames will probably consist of letters and numbers, but other characters are also allowed. Legal characters for filename extensions are the same as those for filenames. Here is a complete list of the characters you can use in filenames and extensions:

A-Z	0-9	\$	&	#
%	'	()	-	@
^	{ }	~	`	!

The syntax for specifying a file and drive is as follows:

[<drive designation:>]<filename>[<.filename extension>]

Remember that brackets indicate optional items. Angle brackets (<>) mean that you supply the text for the item. Note that MS-DOS does not require the drive designation unless you need to indicate which disk to search for a specific file. If you give your filename a filename extension, you must include the extension whenever you enter the filename in an MS-DOS command.

Here are examples of file specifications (the first three include an optional drive specification):

```
B:MYPROG.COB
A:YOURPROG.EXT
A:NEWFILE.
TEXT
```

A file may exist within a subdirectory of the main directory or within a subdirectory of a subdirectory. This concept is described later in this section under "Directories". The syntax for including the name of the path through these subdirectories is described in this section under "Filenames and Paths".

Wild Cards

You can use two special characters (called wild cards) in filenames and extensions: the question mark (?) and the asterisk (*). These special characters give you greater flexibility when using filenames in MS-DOS commands.

The ? Wild Card

A question mark (?) in a filename or filename extension indicates that any character can occupy that position. For example, the MS-DOS command:

```
DIR TEST?RUN.EXE
```

lists all directory entries on the default drive that have 8 characters, begin with TEST, have any next character, end with the letters RUN, and have a filename extension of .EXE. Here are some examples of files that might be listed by the preceding DIR command:

```
TEST1RUN.EXE  
TEST2RUN.EXE  
TEST6RUN.EXE
```

The * Wild Card

An asterisk (*) in a filename or filename extension indicates that any character can occupy that position or any of the remaining positions in the filename or extension. For example:

```
DIR TEST*.EXE
```

lists all directory entries on the default drive with filenames that begin with the characters TEST and have an extension of .EXE. Here are some examples of files that you might list using the DIR TEST*.EXE command:

```
TEST1RUN.EXE  
TEST2RUN.EXE  
TEST6RUN.EXE  
TESTALL.EXE
```

The wild-card designation *.* refers to all files on the disk. Note that this designation can be very powerful and destructive when used in

MS-DOS commands. For example, the command `DEL *.*` deletes all files on the default drive, regardless of filename or extension.

For example, to list the directory entries for all files named `NEWFILE` on drive A (regardless of their filename extensions), type:

DIR A:NEWFILE.*

To list the directory entries for all files with filename extensions of `.TXT` (regardless of their filenames) on the disk in drive B, type:

DIR B:*.TXT

This command is useful if you gave all your text files a filename extension of `.TXT`. By using the `DIR` command with the wild card characters, you can obtain a listing of all your text files even if you do not remember all of their filenames.

Illegal Filenames

MS-DOS treats some device names specially, and certain three-letter names are reserved for the names of these devices. You cannot use these three-letter names as filenames or extensions. You must not name your files any of the following names:

AUX—Used when referring to input from or output to an auxiliary device (such as a printer).

CON—Used when referring to keyboard input or to output to the keyboard console (screen).

LPT or PRN—Used when referring to the printer device.

NUL—Used when you do not want to create a particular file, but the command requires an input or output filename.

Even if you add device designations or filename extensions to these filenames, they remain associated with their listed devices. For example, `A:CON.XXX` still refers to the console and is not the name of a disk file.

Copying Your Files

Just as with paper files, you often need more than one copy of a disk file. The COPY command enables you to copy one or more files to another disk. You can also give the copy a different name if you specify the new name in the COPY command.

The COPY command can also make copies of files on the same disk. In this case, you **must** supply MS-DOS with a different filename or you will overwrite the original file. You cannot make a copy of a file on the same disk unless you specify a different filename for the new copy.

The format of the COPY command is:

```
COPY [d:] filespec [[d:]filespec]
```

For example,

```
COPY A:MYFILE.TXT B:MYFILE.TXT RETURN
```

copies the file MYFILE.TXT on the disk in drive A to a file named MYFILE.TXT on the disk in drive B.

Figure 3-1 illustrates how to copy files from one disk to another:

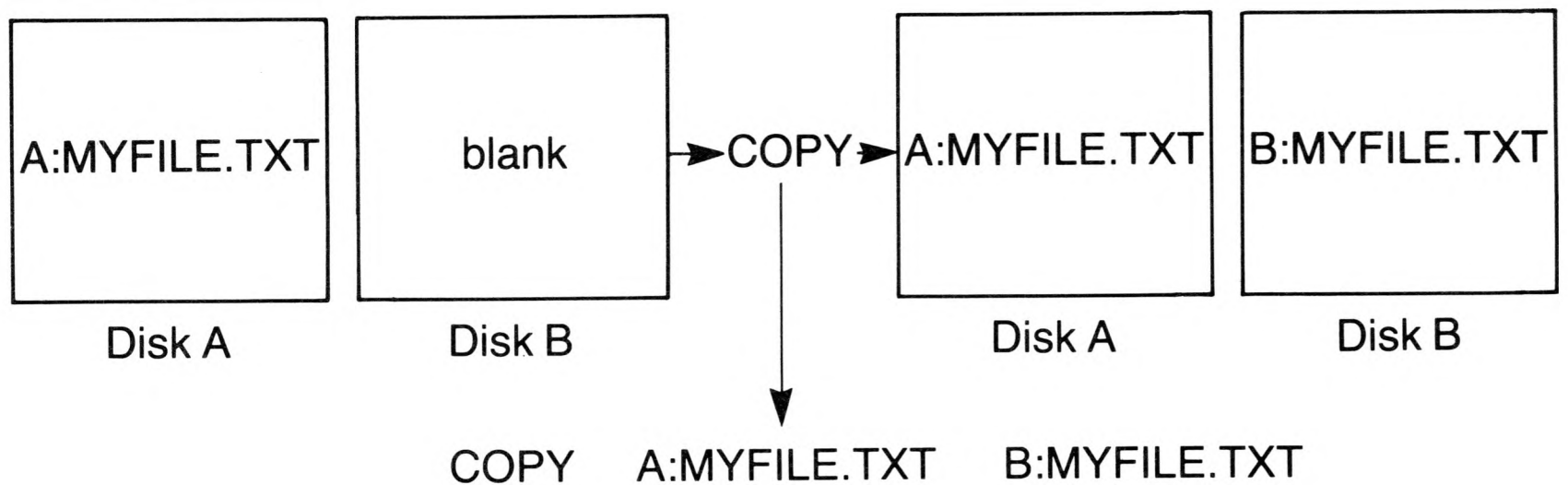


Figure 3-1: Copying files to another disk

If you want to duplicate the file named MYFILE.TXT on the same disk, type:

```
COPY A:MYFILE.TXT A:NEWNAME.TXT RETURN
```

You now have two copies of your file on disk A—one named MYFILE.TXT and the other named NEWNAME.TXT. Figure 3-2 illustrates this example:

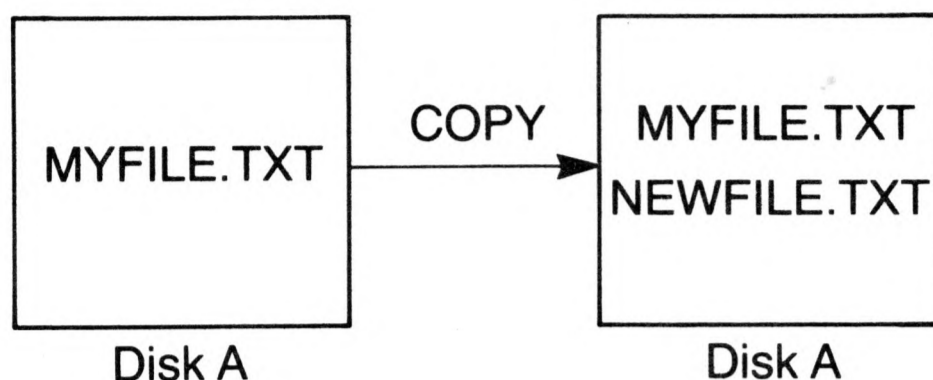


Figure 3-2: Copying files on the same disk

You can also copy all files on a disk to another disk (that is, make a backup copy) with the COPY command. Refer to Section 5, “MS-DOS Commands”, for more information on this process.

Protecting Your Files

MS-DOS is a powerful and useful tool in processing your personal and business information. As with any information system, inadvertent errors may occur and information may be misused. If you are processing information that cannot be replaced or requires a high level of security, you should take steps to protect your data and programs from accidental or unauthorized use, modification, or destruction. Simple measures you can take—such as removing your disks when they are not in use, keeping backup copies of valuable information, and installing your equipment in a secure facility—can help you maintain the integrity of the information in your files.

Directories

As you learned in Section 2, MS-DOS keeps the names of your files in a directory on each disk. The directory also contains information on the size of the files, their locations on the disk, and the dates that they were created and updated.

When there are multiple users on your computer, or when you are working on several different projects, the number of files in the directory

can become large and unwieldy. You may want your own files kept separate from a co-worker's, or you may want to organize your programs into categories that are convenient for you.

In an office, you can separate files by putting them in different filing cabinets; in effect, creating different directories of information. MS-DOS allows you to organize the files on your disks into directories. Directories provide a way of dividing your files into convenient groups of files. For example, you may want all of your accounting programs in one directory and text files in another. Any one directory can contain any reasonable number of files, and it may also contain other directories (referred to as subdirectories). This method of organizing your files is called a hierarchical directory structure.

A hierarchical directory structure can be thought of as a "tree" structure: directories are branches of the tree and files are the leaves, except that the "tree" grows downward; that is, the "root" is at the top. The root is the first level in the directory structure. The root is the directory that MS-DOS automatically creates when you format a disk and start putting files in it. You can create additional directories and subdirectories by following the instructions in Section 4, "Learning About Commands".

The tree or file structure grows as you create new directories for groups of files or for other people on the system. Within each new directory, files can be added, or new subdirectories can be created.

It is possible for you to "travel" around this tree; for instance, it is possible to find any file in the system by starting at the root and traveling down any of the branches to the desired file. Conversely, you can start where you are within the file system and travel upward toward the root.

The filenames discussed earlier in this section are relative to your current directory and do not apply system-wide. Thus, when you start MS-DOS on your computer, you are "in" your directory. Unless you take special action when you create a file, the new file is created in the directory in which you are now working. Users can have files of the same name that are unrelated because each is in a different directory.

Figure 3-3 illustrates a typical hierarchical directory structure.

The ROOT directory is the first level in the directory structure. You can create subdirectories from the ROOT by using the MKDIR command. (Refer to Section 5, "MS-DOS Commands", for information on MKDIR.)

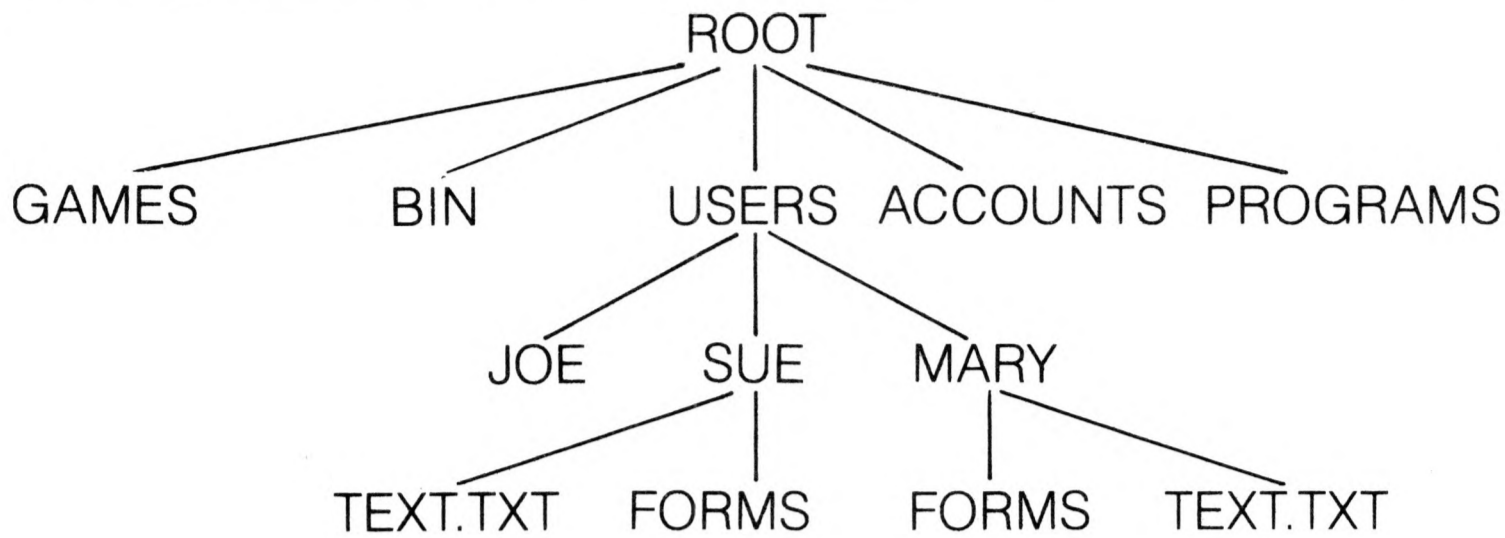


Figure 3-3: A sample hierarchical directory structure

In this example, five subdirectories of ROOT have been created. These directories include:

- A directory of games, named GAMES.
- A directory of all external commands, named BIN. (Refer to Section 4, “Learning About Commands”, for more information on the BIN directory.)
- A USERS directory containing separate subdirectories for all users of the system.
- A directory containing accounting information, named ACCOUNTS.
- A directory of programs, named PROGRAMS.

Joe, Sue, and Mary each have their own directories which are subdirectories of the USERS directory. Sue has a subdirectory under the \USERS\SUE directory named FORMS. Sue and Mary have files in their directories, each named TEXT.TXT. Note that Mary’s text file is unrelated to Sue’s.

This organization of files and directories is not important if you work only with files in your own directory. If you work with someone else or on several projects at one time, however, the hierarchical directory structure becomes extremely useful. For example, you could get a list of the files in Sue’s FORMS directory by typing:

```
DIR \USERS\SUE\FORMS RETURN
```

Note that you use the backward slash mark (\) to separate directories from other directories and files.

To find out what files Mary has in her directory, you could type:

DIR \USERS\MARY RETURN

Filenames and Paths

When you use hierarchical directories, you must tell MS-DOS where the files are located in the directory structure. Both Mary and Sue, for example, have files named TEXT.TXT. Each will have to tell MS-DOS in which directory her file resides if she wants to access it. They can do this by giving MS-DOS a pathname to the file.

Pathnames

A pathname is a sequence of directory names followed by a filespec, each separated from the previous one by a backward slash. A pathname can be optionally preceded by a drive designation.

The syntax of pathnames is:

[<d:>][<directory>]\[<directory...>]\[<filespec>]

If a pathname begins with a backward slash, MS-DOS searches for the file beginning at the root (or top) of the tree. Otherwise, MS-DOS begins at the user's current directory, known as the working directory, and searches downward from there. The pathname of Sue's TEXT.TXT file is \USERS\SUE\TEXT.TXT.

When you are in your working directory, you can use a filename and its corresponding pathname interchangeably. The sample names are:

\	Indicates the root directory.
\PROGRAMS	Sample directory under the root directory containing program files.
\USERS\MARY\FORMS\1A	A typical full pathname. This sample name is a file named 1A in the subdirectory named FORMS belonging to the subdirectory of the USERS subdirectory named MARY. All of these subdirectories are below the root directory.

USERS\SUE	A relative pathname; it names the file or directory SUE in subdirectory USERS of the working directory. If the working directory is the root (\), it names \USERS\SUE.
TEXT.TXT	Name of a file or directory in the working directory.

MS-DOS provides special shorthand notation for the working directory and the parent directory (one level up) of the working directory:

- MS-DOS uses this notation to indicate the name of the working directory in all hierarchical directory listings. MS-DOS automatically creates this entry when a directory is made.
- The shorthand name of the working directory's parent directory. If you type:

DIR ..

then MS-DOS lists the files in the parent directory of your working directory.

If you type:

DIR ..\..

then MS-DOS lists the files in the parent's parent directory.

Pathing And External Commands

External commands reside on disks as program files. They must be read from the disk before they are executed. (For more information on external commands, refer to Section 4, "Learning About Commands".)

When you are working with more than one directory, it is convenient to put all MS-DOS external commands into a separate directory so they do not clutter your other directories. When you issue an external command to MS-DOS, MS-DOS immediately checks your working directory to find that command. You must use the PATH command to tell MS-DOS in which directory these external commands reside.

For example, if you are in a working directory named \BIN\PROG, and all MS-DOS external commands are in \BIN, you must tell MS-DOS to choose the \BIN path to find the FORMAT command. The command:

PATH \BIN

tells MS-DOS to search in your working directory and the \BIN directory for all commands. You only have to specify this path once to MS-DOS during your session. After you enter this command, MS-DOS searches \BIN for the external commands. If you want to know the current path, type the word PATH and MS-DOS displays on the screen the current value of PATH.

For more information on the MS-DOS command PATH, refer to Section 5, "MS-DOS Commands".

Pathing And Internal Commands

Internal commands are the simplest, most commonly used commands. MS-DOS can execute them immediately because they are incorporated into the command processor. (For more information on internal commands, refer to Section 4, "Learning About Commands".)

Some internal commands can use paths. The COPY, DIR, DEL, and TYPE commands have greater flexibility when you specify a pathname after the command.

The syntax of these four commands is as follows:

COPY <pathname pathname>—If the second pathname to COPY is a directory, MS-DOS copies all files into that directory.

DEL <pathname>—If the pathname is a directory, MS-DOS deletes all the files in that directory. Note that MS-DOS displays the prompt "Are you sure (Y/N)?" if you try to delete a path. Type **Y** to complete the command, or type **N** to abort the command.

DIR <pathname>—Displays the directory for a specific path.

TYPE <pathname>—You must specify a file in a path for this command. MS-DOS displays the file on your screen in response to the TYPE pathname command.

Displaying Your Working Directory

MS-DOS executes all commands while you are in your working directory. You can find out the name of the directory you are in by issuing the MS-DOS command CHDIR (Change Directory) with no options. For example, if your current directory is \USERS\JOE and you type:

```
CHDIR RETURN
```

you see:

```
A:\USERS\JOE
```

This message is your current default drive designation plus the working directory (\USERS\JOE).

If you now want to see what is in the \USERS\JOE directory, you can issue the MS-DOS command DIR. The following display is an example of what you might receive from the DIR command for a subdirectory:

```
Volume in drive A has no ID
Directory of A:\USERS\JOE

                <DIR>                8-09-82                10:09a
..                <DIR>                8-09-82                10:09a
TEXT              <DIR>                8-09-82                10:09a
FILE1.COM  5243      8-04-82                9:30a
4 File(s)                8376320 bytes free
```

A volume ID for this disk was not assigned when the disk was formatted. Note that MS-DOS lists both files and directories in response to this command. As you can see, Sue has another directory named TEXT in this tree structure. The '.' indicates the working directory \USERS\JOE, and the '..' is the notation for the parent directory \USERS. FILE1.COM is a file in the \USERS\JOE directory. All of these directories and files reside on the disk in drive A.

Because MS-DOS lists files and directories together (see previous display), you cannot give a subdirectory the same name as a file in that directory. For example, if you have a path \BIN\USERS\SUE where SUE is a subdirectory, you cannot create a file in the USERS directory named SUE.

Creating a Directory

To create a subdirectory in your working directory, use the MKDIR (Make Directory) command. For example, to create a new directory named NEWDIR under your working directory, simply type:

```
MKDIR NEWDIR 
```

After MS-DOS executes this command, a new directory will exist in your tree structure under your working directory. You can also make directories anywhere in the tree structure by specifying MKDIR and then a pathname. MS-DOS automatically creates the . and .. entries in the new directory.

To create files in the new directory, use the MS-DOS line editor, EDLIN, or any other editor. You can also use COPY to put files in the new directory. Section 7, "The Line Editor (EDLIN)", describes how to use EDLIN to create and save files.

Changing Your Working Directory

Changing from your working directory to another directory is very easy in MS-DOS. Simply issue the CHDIR (Change Directory) command and supply a pathname. For example:

```
CHDIR \USERS 
```

changes the working directory to \USERS. You can specify any pathname after the command to "travel" to different branches and leaves of the directory tree. The command "CHDIR .." will always put you in the parent directory of your working directory.

Removing a Directory

To delete a directory in the tree structure, use the MS-DOS RMDIR (Remove Directory) command. For example, to remove the directory NEWDIR from the working directory, type:

```
RMDIR NEWDIR 
```

Note that the directory NEWDIR must be empty except for the . and .. entries before you can remove the directory to prevent you from accidentally deleting files and directories. You can remove any directory

by specifying its pathname. To remove the \BIN\USERS\JOE directory, make sure that it has only the . and .. entries, then type:

RMDIR \BIN\USERS\JOE

To remove all the files in a directory (except for the . and .. entries), type DEL and then the pathname of the directory. For example, to delete all files in the \BIN\USERS\SUE directory, type:

DEL \BIN\USERS\SUE

You cannot delete the . and .. entries. MS-DOS creates them as part of the hierarchical directory structure.

Summary of Commands in this Section

Command	Purpose	Syntax
COPY	Copies files	COPY filespec [filespec]
PATH	Sets MS-DOS search path	PATH [pathname]
CHDIR	Displays working directory; changes directories	CHDIR [pathname]
MKDIR	Makes a new directory	MKDIR [pathname]
RMDIR	Removes a directory	RMDIR [pathname]

In the next section, you learn about MS-DOS commands.

Learning About
Commands

MS-DOS Commands

MS-DOS Editing and
Function Keys

Section 4

Learning About Commands

Introduction

Commands are a way of communicating with the computer. By entering MS-DOS commands at your keyboard, you can ask the system to perform the following useful tasks.

- Compare, copy, display, delete, and rename files.
- Copy and format disks.
- Execute system programs such as EDLIN, as well as your own programs.
- Analyze and list directories.
- Enter the date, the time, and remarks.
- Set various printer and screen options.
- Copy MS-DOS system files to another disk.
- Request MS-DOS to wait for a specific period of time.

Types of MS-DOS Commands

The two types of MS-DOS commands are internal commands and external commands.

Internal Commands

Internal commands are the simplest, most commonly used commands. You cannot see these commands when you list a directory on your MS-DOS disk; they are part of the command processor. When you type these commands, they are executed immediately. Section 5, "MS-DOS Commands", describes the following internal commands (the characters enclosed in parentheses are valid abbreviations for the commands):

BREAK	DEL (ERASE)	MKDIR (MD)	SET
CHDIR (CD)	DIR	PATH	SHIFT
CLS	ECHO	PAUSE	TIME
COPY	EXIT	PROMPT	TYPE
CTTY	FOR	REM	VER
DATE	GOTO	REN (RENAME)	VERIFY
	IF	RMDIR (RD)	VOL

External Commands

External commands reside on disks as program files. MS-DOS must read these commands from disk before they can be executed. If the disk containing the command is not in the drive, then MS-DOS cannot find and execute the command.

External commands always have filenames with a filename extension of .COM, .EXE or .BAT. For example, programs such as FORMAT.COM and COMP.COM are external commands. Because all external commands reside on disk, you can create commands and add them to the system. Programs that you create with most languages (including assembly language) will be .EXE (executable) files.

When you enter an external command, do not include its filename extension. Section 5 describes the following external commands:

CHKDSK	MORE
DISKCOPY	PRINT
EXE2BIN	RECOVER
FIND	SORT
FORMAT	SYS
MODE	

Command Options

You can include options in your MS-DOS commands to specify additional information to the system. If you do not include some options, MS-DOS provides a default value. Refer to individual command descriptions in Section 5, "MS-DOS Commands", for the default values.

Here is the format of all MS-DOS commands:

Command [options...]

where:

- | | |
|-----------|--|
| d: | Refers to the disk drive designation. |
| filename | Refers to any valid name for a disk file, including an optional filename extension. The filename option does not refer to a device or to a disk drive designation. |
| .ext | Refers to an optional filename extension consisting of a period and 1 to 3 characters. When used, filename extensions immediately follow filenames. |
| filespec | Refers to a filename and an optional three-letter filename extension in the following format:

<filename>[<.ext>] |
| pathname | Refers to a pathname or filename in the following format:

[<directory>]\[<directory...>]\[<filespec>] |
| switches | Options that control MS-DOS commands. They are preceded by a slash (for example, /P). |
| arguments | Provide more information to MS-DOS commands. You usually choose between arguments, such as ON or OFF. |

Information Common to All MS-DOS Commands

The following information applies to all MS-DOS commands:

1. One or more options follow most commands.
2. You may enter commands and options in uppercase or lowercase letters, or a combination of the two.
3. You must use delimiters to separate commands and options. The space and the comma are the two most commonly used delimiters. For example:

```
DEL MYFILE.OLD NEWFILE.TXT  
RENAME,THISFILE THATFILE
```

You can also use the semicolon (;), the equal sign (=), or the tab key as delimiters in MS-DOS commands.

This manual uses a space as the delimiter in commands.

4. Do not separate a file specification with delimiters, since the colon and the period already serve as delimiters.
5. When instructions say "Press any key", you can press any alphabetic (A-Z) or numeric (0-9) key.
6. You must include the filename extension when referring to a file that already has a filename extension.
7. You can abort commands while they are running by pressing the CTRL-C sequence of keys.
8. Commands take effect only after you press the RETURN key.
9. You cannot use wild card characters (* and ?) in the name of any command. Also, you cannot use device names such as PRN or CON as a command name.

10. When commands produce a large amount of output on the screen, the display automatically scrolls to the next screen. You can press CTRL-S to suspend the display. Press any key to resume the display on the screen.

Note: Do not press any other key before pressing CTRL-S, or the scrolling will continue.

11. You can use MS-DOS editing and function keys when entering commands. Refer to Section 6, "MS-DOS Editing and Function Keys", for a complete description of these keys.

12. The prompt from the command processor is the default drive designation plus a greater-than sign; for example, A>.

13. This manual refers to disk drives as source drives and destination drives. A source drive is the drive from which you are transferring information. A destination drive is the drive to which you are transferring information.

Batch Processing

Often you may find yourself typing the same sequence of commands over and over to perform some commonly used task. With MS-DOS, you can put the command sequence into a special file called a batch file, and execute the entire sequence simply by typing the name of the batch file. MS-DOS processes "batches" of your commands in such files as if you typed them on your keyboard. You must include a .BAT extension with each batch file. To execute the batch file, type the filename without its extension.

You can create a batch file by using the Line Editor (EDLIN) or by typing the COPY command. See "Creating an AUTOEXEC.BAT File" later in this section for more information on using the COPY command to create a batch file.

Two MS-DOS commands are available for use expressly in batch files: REM and PAUSE. REM permits you to include remarks and comments in your batch files without these remarks being executed as commands. PAUSE prompts you with an optional message and permits you to either continue or abort the batch process at a given point. Section 5, "MS-DOS Commands", describes REM and PAUSE in detail.

Batch processing is useful if you want to execute several MS-DOS commands with one batch command, such as when you format and check a new disk. For example, a batch file for this purpose might look like this:

```
1: REM This is a file to check new disks
2: REM It is named NEWDISK.BAT
3: PAUSE Insert new disk in drive B:
4: FORMAT B:
5: DIR B:
6: CHKDSK B:
```

To execute this .BAT file, simply type the filename without the .BAT extension:

```
NEWDISK 
```

The result is the same as if you entered each of the lines in the .BAT file on the keyboard as individual commands.

Figure 4-1 illustrates the three steps used to write, save, and execute an MS-DOS batch file.

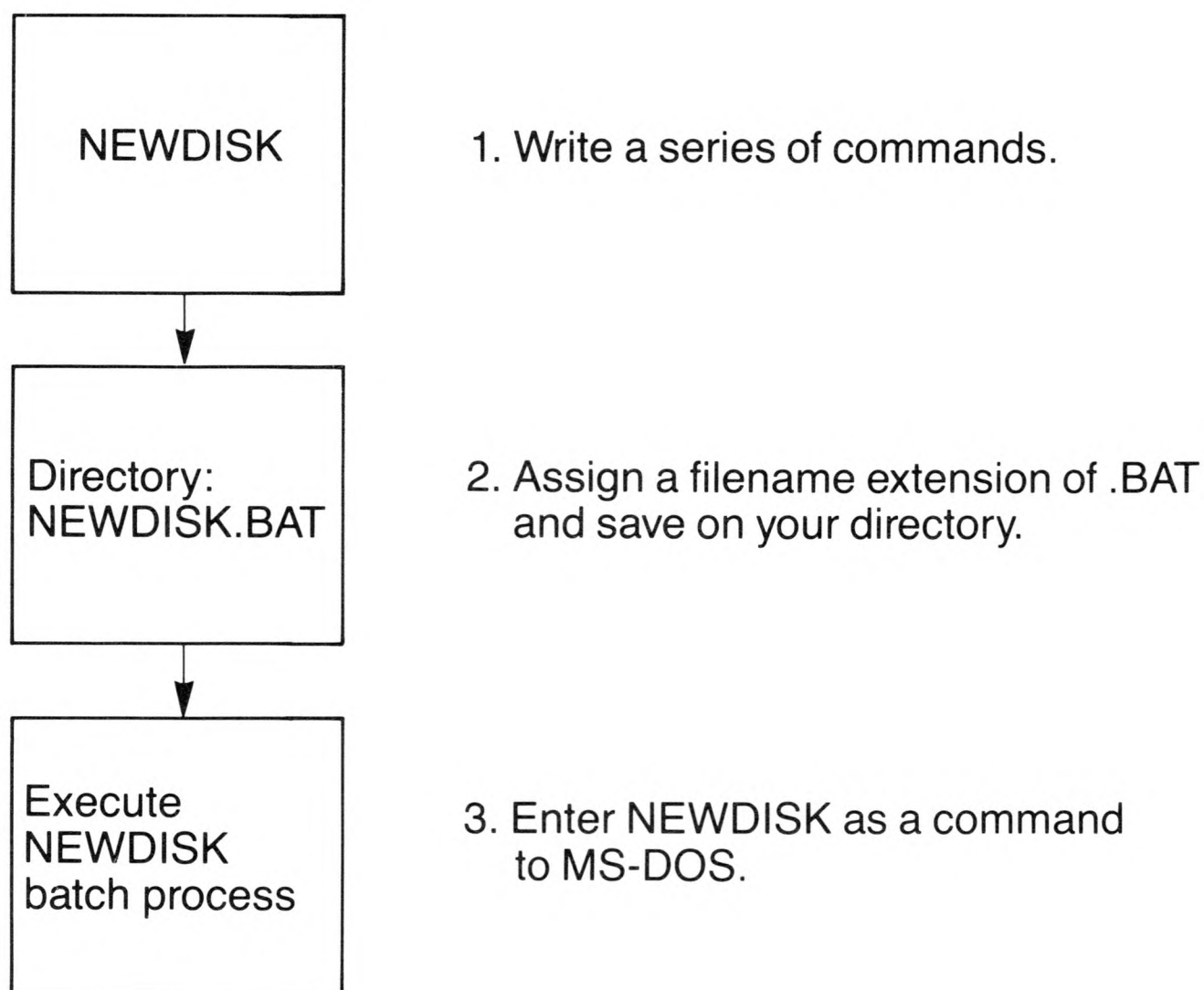


Figure 4-1: MS-DOS batch file steps

The following list contains information that you should read before you execute a batch process with MS-DOS:

1. Do not enter the filename BATCH (unless the name of the file you want to execute is BATCH.BAT).
2. Enter only the filename to execute the batch file. Do not enter the filename extension.
3. MS-DOS executes the commands in the file named <filename>.BAT.
4. If you press CTRL-C while in batch mode, this prompt appears:

Terminate batch job (Y/N)?

If you press **Y**, MS-DOS skips the remainder of the commands in the batch file and displays the system prompt.

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

5. If you remove the disk containing a batch file being executed, MS-DOS prompts you to insert it again so that it can read the next command.
6. Because the last command in a batch file may be the name of another batch file, you can call one batch file from another when the first is finished.

The AUTOEXEC.BAT File

As discussed in Section 2, an AUTOEXEC.BAT file enables you to automatically execute programs when you start MS-DOS. At this time, the command processor searches the MS-DOS disk for a file named AUTOEXEC.BAT which is a batch file that MS-DOS executes automatically each time you start the system.

Automatic Program Execution is useful when you want to run a specific program under MS-DOS, and when you want MS-DOS to execute a batch program automatically each time you start the system. You can avoid loading two separate disks to perform either of these tasks by using an AUTOEXEC.BAT file.

If MS-DOS finds the AUTOEXEC.BAT file, the command processor executes the file immediately and bypasses the date and time prompts.

If MS-DOS does not find an AUTOEXEC.BAT file when you first load the MS-DOS disk, then it issues the date and time prompts. Figure 4-2 illustrates how MS-DOS uses the AUTOEXEC.BAT file.

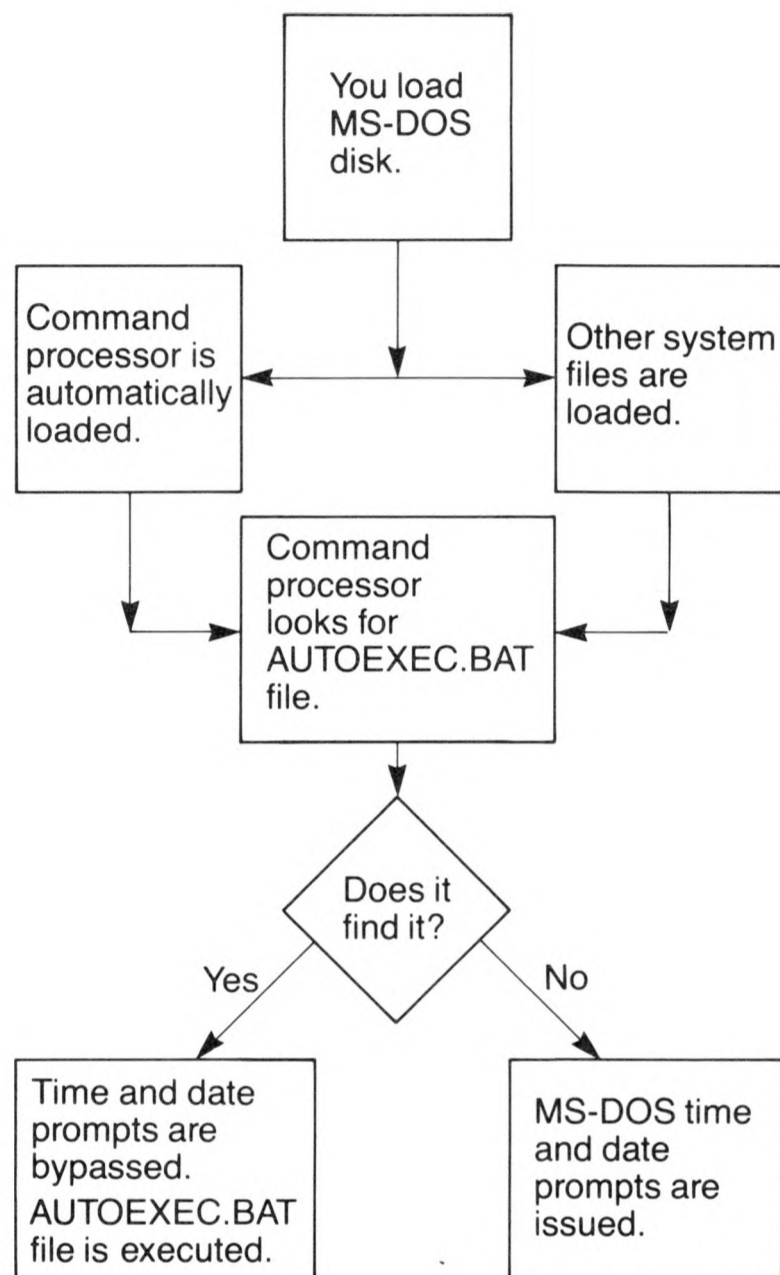


Figure 4-2: How MS-DOS uses the AUTOEXEC.BAT file

Creating an AUTOEXEC.BAT File

If, for example, you want to automatically load BASIC and run a program called MENU each time you start MS-DOS, you can create an AUTOEXEC.BAT file as follows:

1. Type:

```
COPY CON AUTOEXEC.BAT RETURN
```

This statement tells MS-DOS to copy the information from the console (keyboard) into the AUTOEXEC.BAT file. Note that you must create the AUTOEXEC.BAT file in the root directory of your MS-DOS disk.

2. Now type:

DIR

This statement goes into the AUTOEXEC.BAT file. It tells MS-DOS to display a directory of the system disk whenever you start MS-DOS.

3. Press the CTRL-Z sequence, then press the RETURN key to put the command DIR in the AUTOEXEC.BAT file.
4. The DIR program now runs automatically whenever you start MS-DOS using a disk containing this AUTOEXEC.BAT file.

To run your own BASIC program, enter the name of your program in place of DIR in the second line of the example. You can enter any MS-DOS command or series of commands in the AUTOEXEC.BAT file.

Note: Remember that if you use an AUTOEXEC.BAT file, MS-DOS does not prompt you for a current date and time unless you include the DATE and TIME commands in the AUTOEXEC.BAT file. It is strongly recommended that you include these two commands in your AUTOEXEC.BAT file, because MS-DOS uses this information to keep your directory current.

Creating a .BAT File with Replaceable Parameters

There may be times when you want to create an application program and run it with different sets of data. These data may be stored in various MS-DOS files.

When used in MS-DOS commands, a parameter is an option that you define. With MS-DOS, you can create a batch (.BAT) file with dummy (replaceable) parameters. You can then supply the real values for these parameters, named %0-%9, when the batch file is executed.

For example, when you type the command line COPY CON MYFILE.BAT, MS-DOS copies the next lines you type from the keyboard to a file named MYFILE.BAT on the default drive:

```
A>>COPY CON MYFILE.BAT   
COPY %1.MAC %2.MAC   
TYPE %2.PRN   
TYPE %0.BAT 
```

Press the CTRL-Z sequence, and then press RETURN. MS-DOS responds with this message:

```
1 File(s) copied
A>_
```

The file MYFILE.BAT, which consists of three commands, now resides on the disk in the default drive, drive A.

MS-DOS replaces the dummy parameters %1 and %2 sequentially by the parameters you supply when you execute the file. MS-DOS always replaces the dummy parameter %0 with the drive designator, if specified, and the filename of the batch file (for example, MYFILE).

Note the following points:

1. You can normally specify up to 10 dummy parameters (%0-%9) in a batch file. Refer to the SHIFT command in Section 5, "MS-DOS Commands", if you wish to specify more than 10 parameters.
2. If you use the percent sign as part of a filename within a batch file, you must type it twice. For example, to specify the file ABC%.EXE, you must type it as ABC%%.EXE in the batch file.

Executing a .BAT File

To execute the batch file MYFILE.BAT and to specify the parameters that will replace the dummy parameters, you must enter the batch filename (without its extension) followed by the parameters you want MS-DOS to substitute for %1, %2, and so on.

Remember that the file MYFILE.BAT consists of three lines:

```
COPY %1.MAC %2.MAC
TYPE %2.PRN
TYPE %0.BAT
```

To execute the MYFILE batch process, type:

```
MYFILE A:PROG1 B:PROG2 RETURN
```

The following list illustrates how MS-DOS replaces each of the preceding parameters:

Batch Filename:	MYFILE.BAT
Parameter 1 (%0):	MYFILE
Parameter 2 (%1):	A: PROG1
Parameter 3 (%2):	B: PROG2

Remember that MS-DOS always replaces the dummy parameter %0 with the drive designator (if specified) and the filename of the batch file.

The result is the same as if you typed each of the commands in MYFILE with their parameters, as follows:

```
COPY A:PROG1.MAC B:PROG2.MAC   
TYPE B:PROG2.PRN   
TYPE MYFILE.BAT 
```

Input and Output

MS-DOS always assumes that input comes from the keyboard and output goes to the screen. However, you can redirect the flow of command input and output. Input can come from a file rather than a keyboard, and output can go to a file or to a line printer instead of to the screen. In addition, you can create “pipes” that enable output from one command to become the input to another. The following sections discuss redirection, filters, and pipes.

Redirecting Your Output

Most commands produce output that appears on the screen. You can send this information to a file by using a greater-than sign (>) in your command. For example, the command:

```
DIR 
```

displays a directory listing of the disk in the default drive on the screen. The same command can send this output to a file named MYFILES by designating the output file on the command line:

```
DIR >MYFILES 
```

If the file MYFILES does not already exist, MS-DOS creates it and stores your directory listing in it. If MYFILES already exists, MS-DOS overwrites what is in the file with the new data.

If you want to append your directory or a file to another file (instead of replacing the entire file), use two greater-than signs (>>) to tell MS-DOS to append the output of the command (such as a directory listing) to the end of a specified file. The command:

DIR >>MYFILES

appends your directory listing to a currently existing file named MYFILES. If MYFILES does not exist, MS-DOS creates it.

It is often useful to have input for a command come from a file rather than from your keyboard. This is possible in MS-DOS by using a less-than sign (<) in your command. For example, the command:

SORT <NAMES >LIST1

sorts the file NAMES and sends the sorted output to a file named LIST1.

Filters

A filter is a command that reads your input, transforms it in some way, and then sends it to your screen or to a file. In this way, the data is said to have been “filtered” by the program. Since you can put filters together in many different ways, a few filters can take the place of a large number of specific commands.

The names of the MS-DOS filters are FIND, MORE, and SORT. Their functions are as follows:

FIND—Searches for a constant string of text in a file.

MORE—Takes standard keyboard output and displays it, one screen at a time.

SORT—Sorts text.

You can find out how to use these filters in the following discussion.

Command Piping

If you want to give more than one command to the system at a time, you can “pipe” commands to MS-DOS. For example, you may occasionally need to send the output of one program as the input to another program. A typical case is a program that produces output in columns. It is desirable to have this columnar output sorted.

You can specify command piping by separating commands with the pipe separator, which is the vertical bar symbol (|). For example, the command:

```
DIR | SORT 
```

gives you an alphabetically sorted listing of your directory. The vertical bar causes all output generated by the left side of the bar to be sent to the right side of the bar for processing.

You can also use piping when you want to send output to a file. If you want your directory sorted and sent to a new file (for example, DIREC.FIL), you could type:

```
DIR | SORT >DIREC.FIL 
```

MS-DOS creates a file named DIREC.FIL on your default drive. DIREC.FIL contains a sorted listing of the directory on the default drive, since you specified no other drive in the command. To specify a drive other than the default drive, type:

```
DIR | SORT >B:DIREC.FIL 
```

This command sends the sorted data to a file named DIREC.FIL on drive B.

A pipeline may consist of more than two commands. For example:

```
DIR | SORT | MORE 
```

sorts your directory, shows it to you one screen at a time, and puts —MORE— at the bottom of your screen when there is more output which MS-DOS has not yet displayed.

You will find many uses for piping commands and filters.

Summary of Commands in this Section

Command	Purpose	Syntax
REM	Adds comment line for batch files	REM [remark]
PAUSE	Suspends execution of a batch file	PAUSE [comment]
FIND	Searches for a string of text	FIND string [filespec]
MORE	Pages through a file 23 lines at a time	MORE
SORT	Sorts text	SORT

You can find more information on using these filters in the next section, "MS-DOS Commands".

Section 5

MS-DOS Commands

Documentation Conventions for Commands

The following notation shows how to format MS-DOS commands:

1. You must enter any words that are shown in capital letters. These words are called keywords and must be entered exactly as shown. You can enter these keywords in any combination of uppercase or lowercase letters; MS-DOS converts all keywords to uppercase letters.
2. You supply the text for any items enclosed in angle brackets (<>). For example, you should enter the name of your file when <filename> is shown in the format.
3. Items in square brackets ([]) are optional. If you wish to include optional information, do not include the square brackets, only the information within the brackets.
4. An ellipsis (...) indicates that you may repeat an item as many times as you want.
5. A vertical bar represents an OR. When you see this OR bar between two or more parameters, enter only one of the parameters. For example, ON|OFF means to enter either ON or OFF.

-
6. You must include all punctuation where shown (except square brackets and vertical bars), such as commas, equal signs, question marks, colons, and slashes.
-

MS-DOS Command Summary

Note: Users of single-drive systems should refer to Appendix A for the additional procedures required to execute many of the following commands.

This section describes the following MS-DOS commands. Note that synonyms for commands are enclosed in parentheses.

BREAK	Sets CTRL-C check
CHDIR	Changes directories, prints the working directory (CD)
CHKDSK	Scans the directory of the default or designated drive and checks for consistency
CLS	Clears the screen
COPY	Copies the file(s) specified
CTTY	Changes console TTY
DATE	Displays and sets the date
DEL	Deletes file(s) specified (ERASE)
DIR	Lists requested directory entries
DISKCOPY	Copies disks
EXE2BIN	Converts executable files to a binary format
EXIT	Exits from the command and returns to a lower level

FIND	Searches for a constant string of text
FORMAT	Formats a disk to receive MS-DOS files
MKDIR	Makes a directory (MD)
MODE	Sets operation parameters
MORE	Displays output one screen at a time
PATH	Sets a command search path
PRINT	Background print feature
PROMPT	Designates the command prompt
RECOVER	Recovers a bad disk
REM	Displays a comment in a batch file
REN	Renames the first file as the second file (RENAME)
RMDIR	Removes a directory (RD)
SET	Sets one string value to another
SORT	Sorts data alphabetically, forward or backward
SYS	Transfers MS-DOS system files from drive A to the drive specified
TIME	Displays and sets the time
TYPE	Displays the contents of the specified file
VER	Prints the MS-DOS version number
VERIFY	Verifies information written to a disk
VOL	Prints the volume identification number

Batch Commands (Command Extensions) Summary

ECHO	Turns batch file echo feature on/off
FOR	Batch command extension
GOTO	Batch command extension
IF	Batch command extension
PAUSE	Pauses for input in a batch file
SHIFT	Increases number of replaceable parameters in batch process

MS-DOS Command Descriptions

The following commands control MS-DOS operation.

NAME: BREAK**TYPE:** Internal**PURPOSE:**

Sets CTRL-C check.

SYNTAX:

BREAK [ON|OFF]

NOTES:

If you are running an application program that uses CTRL-C function keys, you will want to turn off the MS-DOS CTRL-C function so that when you press the CTRL-C sequence you affect your program and not the operating system. Specify BREAK OFF to turn off CTRL-C and BREAK ON when you have finished running your application program and are using MS-DOS.

NAME: CHDIR (CHANGE DIRECTORY)**TYPE:** Internal**SYNONYM:**

CD

PURPOSE:

Changes the directory to a different path; displays the current (working) directory.

SYNTAX:

CHDIR [pathname]

NOTES:

If your working directory is \BIN\USERS\JOE, for example, and you want to change your path to another directory (such as \BIN\USERS\JOE\FORMS), type:

```
CHDIR\BIN\USERS\JOE\FORMS 
```

and MS-DOS puts you in the new directory. A notation is also available with this command:

```
CHDIR ..
```

This command always puts you in the parent directory of your working directory.

CHDIR used without a pathname displays your working directory. If your working directory is \BIN\USERS\JOE on drive B and you type **CHDIR**

, MS-DOS displays:

```
B: \BIN\USERS\JOE
```

This command is useful if you forget the name of your working directory.

Note: The notation CHDIR.. cannot be used if your working directory is more than eight levels below the root directory.

NAME: CHKDSK (CHECK DISK)**TYPE:** External**PURPOSE:**

Scans the directory of the specified disk drive and checks it for consistency.

SYNTAX:

CHKDSK [d:] <filespec> [/F] [/V]

NOTES:

You should run CHKDSK occasionally on each disk to check for errors in the directory. CHKDSK displays on the screen a status report for the specified disk. If MS-DOS finds any errors, CHKDSK displays any error messages before the status report.

Here is a sample status report:

160256	bytes total disk space
8192	bytes in 2 hidden files
512	bytes in 2 directories
30720	bytes in 8 user files
121344	bytes available on disk
65536	bytes total memory
53152	bytes free

CHKDSK does not correct the errors found in your directory unless you specify the /F (fix) switch. Typing /V causes CHKDSK to display messages while it is running.

You can redirect the output from CHKDSK to a file by typing:

CHKDSK A:>filespec

MS-DOS then sends the error messages to the filespec specified. Do not use the /F switch if you redirect CHKDSK output.

MS-DOS corrects the following errors automatically if you specify the /F switch:

- Invalid drive specification
- Invalid parameter
- Invalid sub-directory entry

Cannot CHDIR to <filespec>, Tree past this point not processed

First cluster number is invalid entry truncated

Allocation error, size adjusted

Has invalid cluster, file truncated

Disk error reading FAT

Disk error writing FAT

<filespec> contains non-contiguous blocks

All specified file(s) are contiguous

You must specify the /F switch if you want the errors corrected by CHKDSK. The following message appears if MS-DOS finds errors which it could have corrected if you had used the /F switch:

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

You must correct the following errors returned by CHKDSK, even if you specified the /F switch.

Incorrect DOS version

You cannot run CHKDSK on versions of MS-DOS that are not 2.0 or higher.

Insufficient memory
Processing cannot continue

There is not enough memory in your machine to process CHKDSK for this disk. You must obtain more memory to run CHKDSK.

Invalid current directory
Processing cannot continue
Restart the system and re-run CHKDSK.
Cannot CHDIR to root
Processing cannot continue

CHKDSK (CHECK DISK), *cont.*

The disk you are checking is bad. Try restarting MS-DOS and RECOVER the disk.

<filename> is cross linked on cluster

Make a copy of the file you want to keep, and then delete both files that are cross linked.

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

If you respond **Y** to this prompt, CHKDSK creates a directory entry and a file for you to resolve this problem (files created by CHKDSK are named FILEnnnnnnnn).

CHKDSK then displays:

x bytes disk space freed

If you respond **N** to this prompt and have not specified the /F switch, CHKDSK frees the clusters and displays:

x bytes disk space would be freed

Probable non-DOS disk
Continue (Y/N)?

The disk you are using is a non-DOS disk. You must indicate whether or not you want CHKDSK to continue processing.

Insufficient room in root directory
Erase files in root and repeat CHKDSK

CHKDSK cannot process until you delete files in the root directory.

Unrecoverable error in directory
Convert directory to file (Y/N)?

If you respond **Y** to this prompt, CHKDSK converts the bad directory into a file. You can then fix the directory yourself or delete it.

NAME: CLS

TYPE: Internal

PURPOSE:

Clears the screen.

SYNTAX:

CLS

NOTES:

The CLS command causes MS-DOS to send the ANSI escape sequence ESC[2J (which clears your screen) to your console.

NAME: COPY**TYPE:** Internal**PURPOSE:**

Copies one or more files to another disk or device. If you prefer, you can give the copies different names. This command can also copy files on the same disk.

SYNTAX:

COPY [d:] [pathname] <filespec> [d:] [pathname] [filespec] [/V]

NOTES:

If you do not include the second drive or filespec option, MS-DOS makes the copy on the default drive and gives it the same name as the original file (first filespec option). If the first filespec is on the default drive and you do not specify the second filespec, MS-DOS aborts the COPY. MS-DOS does not allow you to copy files onto themselves. If you try this operation, MS-DOS prints the following error message:

```
File cannot be copied onto itself
0 File(s) copied
```

The second option may take three forms:

1. If the second option is a drive designation (d:) only, MS-DOS copies the original file with the original filespec to the designated drive.
2. If the second option is a filespec only (with no drive specification), MS-DOS copies the original file to a file on the default drive with the filespec specified.
3. If the second option is a filespec with a drive specification, MS-DOS copies the original file to a file on the specified drive with the filespec specified.

The /V switch causes MS-DOS to verify that the sectors written on the destination disk are recorded properly. Although there are rarely recording errors when you run COPY, you can verify that the disk drive has correctly recorded critical data. This option causes the COPY command to run more slowly because MS-DOS must check each entry recorded on the disk.

The COPY command also allows file concatenation (joining) while copying. Concatenation is accomplished by simply listing any number of files as options to COPY, separated by a plus sign (+).

For example:

COPY A.XYZ + B.COM + B:C.TXT BIGFILE.CRP

This command concatenates files named A.XYZ, B.COM, and B:C.TXT and places them in the file called BIGFILE.CRP on the default drive.

To combine several files using wild cards into one file, you could type:

COPY *.LST COMBIN.PRN

This command would take all files with a filename extension of .LST and combine them into a file named COMBIN.PRN.

In the following example, each file that matches *.LST is combined with the corresponding .REF file. The result is a file with the same filename but with the extension .PRN. Thus, FILE1.LST is combined with FILE1.REF to form FILE1.PRN, then XYZ.LST with XYZ.REF to form XYZ.PRN, and so on.

COPY *.LST + *.REF *.PRN

The following COPY command combines all files matching *.LST, then all files matching *.REF, into one file named COMBIN.PRN:

COPY *.LST + *.REF COMBIN.PRN

Do not enter a concatenation COPY command where one of the source filenames has the same extension as the destination. For example, the following command is an error if ALL.LST already exists:

COPY *.LST ALL.LST

The error would not be detected, however, until ALL.LST is appended. At this point it could have already been destroyed.

COPY compares the filespec of the input file with the filespec of the destination. If they are the same, that one input file is skipped, and the error message "Content of destination lost before copy" is printed. Further concatenation proceeds normally. This process allows "summing" files, as follows:

COPY ALL.LST + *.LST

COPY, *cont.*

This command appends all *.LST files, except ALL.LST itself, to ALL.LST. This command does not produce an error message and is the correct way to append files using the COPY command.

NAME: CTTY**TYPE:** Internal**PURPOSE:**

Enables you to change the device from which you issue commands (TTY represents the console).

SYNTAX:

CTTY <device>

NOTES:

The device is the device from which you are giving commands to MS-DOS. This command is useful if you want to change the device on which you are working. The command:

CTTY AUX

moves all command I/O (input/output) from the current device (the console) to the AUX port, such as a printer. The command:

CTTY CON

moves I/O back to the original device (here, the console). Refer to Section 3, "More About Files", for a list of valid device names to use with the CTTY command.

NAME: DATE**TYPE:** Internal**PURPOSE:**

Enters or changes the date known to the system. This date is recorded in the directory for any files you create or alter.

You can change the date from your keyboard or from a batch file. (MS-DOS does not display a prompt for the date if you use an AUTOEXEC.BAT file, so you may want to include a DATE command in that file.)

SYNTAX:

DATE [<mm>-<dd>-<yy>]

NOTES:

If you type DATE, DATE responds with the message:

```
Current date is <mm>-<dd>-<yy>
Enter new date: __
```

Press RETURN if you do not want to change the date shown.

You can also type a particular date after the DATE command, as in:

```
DATE 3-9-81 RETURN
```

In this case, you do not have to wait for the "Enter new date:" prompt.

Use only numerals to enter the new date; letters are not permitted. The permitted values are:

<mm>	= 1 to 12
<dd>	= 1 to 31
<yy>	= 80 to 99 or 1980 to 2099

You must separate the date, month, and year entries by hyphens (-) or slashes (/). MS-DOS is programmed to change months and years correctly, including leap years.

If the options or separators are invalid, DATE displays the message:

```
Invalid date
Enter new date: __
```

DATE then waits for you to enter a valid date.

NAME: DEL (DELETE)

TYPE: Internal

SYNONYM:

ERASE

PURPOSE:

Deletes all files with the designated filespec.

SYNTAX:

DEL [d:] [pathname][filespec]

NOTES:

If the filespec is *.* , the prompt "Are you sure?" appears. If you type **Y** or **y** as a response, then MS-DOS deletes all files as requested. You can also type ERASE for the DEL command.

NAME: DIR (DIRECTORY)

TYPE: Internal

PURPOSE:

Lists the files in a directory.

SYNTAX:

DIR [d:] [pathname][filespec][/P][/W]

NOTES:

If you type DIR, MS-DOS lists all directory entries in the current directory on the default drive. If you give only the drive specification (DIR d:), MS-DOS lists all entries in the current directory on the disk in the specified drive. If you enter only a filename with no extension (DIR filename), then MS-DOS lists all files with the designated filename in the current directory on the disk in the default drive. If you designate a file specification (for example, DIR d:filename.ext), MS-DOS lists all files with the filespec specified in the current directory on the disk in the drive specified. In all cases, MS-DOS lists files with their size in bytes and with the time and date of their last modification.

You may specify a path in the DIR command. This causes MS-DOS to list all files in the specified path, qualified by drive and filename as above.

You may use the wild card characters ? and * in the filespec option. Note that for your convenience, the following DIR commands are equivalent:

Command	Equivalent
DIR	DIR *.*
DIR FILENAME	DIR FILENAME.*
DIR .EXT	DIR *.EXT

You may specify two switches with DIR. The /P switch selects Page Mode. With /P, display of the directory pauses after it fills the screen. To view the next page of the directory, press any key.

The /W switch selects Wide Display, which places five filenames on each line. With /W, MS-DOS displays only filenames without other file information.

NAME: DISKCOPY**TYPE:** External**PURPOSE:**

Copies the contents of the disk in the source drive to the disk in the destination drive.

SYNTAX:

DISKCOPY [d:] [d:]

NOTES:

The first option you specify is the source drive. The second option is the destination drive.

You must format the disk in the destination drive prior to using DISKCOPY.

You can specify the same drives or different drives. If the drives designated are the same, MS-DOS performs a single-drive copy operation. The system prompts you to insert the disks at the appropriate times. DISKCOPY waits for you to press any key before continuing.

After copying, DISKCOPY prompts:

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

If you press **Y**, MS-DOS performs the next copy on the same drives that you originally specified, after it prompts you to insert the proper disks.

To end the DISKCOPY program, press **N**.

1. If you omit both options, MS-DOS performs a single-drive copy operation on the default drive.
2. If you omit the second option, MS-DOS uses the default drive as the destination drive.
3. Both disks must have the same number of physical sectors, which must be the same size.
4. Disks that have had a lot of file creation and deletion activity become fragmented, because disk space is not allocated sequentially. MS-DOS allocates the first free sector it finds, regardless of its location on the disk.

DISKCOPY, cont.

A fragmented disk can cause poor performance due to delays involved in finding, reading, or writing a file. If so, you should use the COPY command, instead of DISKCOPY, to copy your disk and eliminate the fragmentation.

For example:

COPY A:*. * B:

copies all files from the disk in drive A to the disk in drive B.

5. DISKCOPY automatically determines the number of sides to copy, based on the source drive and disk.
6. If MS-DOS encounters disk errors during a DISKCOPY, it displays this message:

DISK error while reading drive d:, Abort, Ignore, Retry?

Refer to Appendix B, "Disk Errors", for information on this error message.

NAME: EXE2BIN**TYPE:** External**PURPOSE:**

Converts .EXE (executable) files to binary format, which saves disk space and enables faster program loading.

SYNTAX:

EXE2BIN [d:] <filespec>[d:] <filename> [<.ext>]]

NOTES:

This command is useful only if you want to convert .EXE files to binary format. The file named by filespec is the input file. If you do not specify an extension, it defaults to .EXE. MS-DOS converts the input file to .COM file format (memory image of the program) and places it in the output file. If you do not specify a drive, MS-DOS uses the drive of the input file. If you do not specify an output filename, MS-DOS uses the input filename. If you do not specify a filename extension in the output filespec, MS-DOS gives the new file an extension of .BIN.

The input 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.

Two kinds of conversions are possible, depending on whether you specify the initial CS:IP (Code Segment:Instruction Pointer) in the .EXE file:

1. If you do not specify CS:IP in the .EXE file, MS-DOS assumes a pure binary conversion. If segment fixups are necessary (that is, the program contains instructions requiring segment relocation), MS-DOS prompts you to supply the fixup value. This value is the absolute segment at which MS-DOS is to load the program. The resulting program will be usable only when loaded at the absolute memory address specified by a user application. The command processor will not be capable of properly loading the program.
2. If you specify CS:IP as 0000:100H, MS-DOS assumes that the file is to be run as a .COM file with the location pointer set at 100H by the assembler statement ORG; MS-DOS deletes the first 100H bytes of the file. No segment fixups are allowed, as .COM files must be segment relocatable; that is, they must assume the entry conditions explained in the Macro Assembler Manual. Once the conversion is

EXE2BIN, *cont.*

complete, you may rename the resulting file with a .COM extension. Then the command processor can load and execute the program in the same way as the .COM programs supplied on your MS-DOS disk.

If CS:IP does not meet either of these criteria, or if it meets the .COM file criterion but has segment fixups, MS-DOS displays the following message:

File cannot be converted

MS-DOS also displays this message if the file is not a valid executable file.

If EXE2BIN finds an error, MS-DOS displays one or more of the following error messages:

File not found

The file is not on the disk specified.

Insufficient memory

There is not enough memory to run EXE2BIN.

File creation error

EXE2BIN cannot create the output file. Run CHKDSK to determine if the directory is full, or if some other condition caused the error.

Insufficient disk space

There is not enough disk space to create a new file.

Fixups needed—base segment (hex):

The source (.EXE) file contained information indicating that a load segment is required for the file. Specify the absolute segment address at which you want to locate the finished module.

File cannot be converted

The input file is not in the correct format.

WARNING—Read error on EXE file.

The amount read is less than the size in the header. This is a warning message only.

NAME: EXIT**TYPE:** Internal**PURPOSE:**

Exits the program COMMAND.COM (the command processor) and returns to a previous level, if one exists.

SYNTAX:

EXIT

NAME: FIND**TYPE:** External**PURPOSE:**

Searches for a specific string of text in a file or files.

SYNTAX:

FIND [/V] [/C] [/N] <string> [<filespec...>]

NOTES:

FIND is a filter that takes as options a string and a series of filespecs. It displays all lines that contain a specified string from the files specified in the command line.

If you do not specify any files, FIND will take the input on the screen and display all lines that contain the specified string.

Switches for FIND are:

- /V causes FIND to display all lines not containing the specified string.
- /C causes FIND to print only the count of lines that contain a match in each of the files. (Overrides other options.)
- /N causes each line to be preceded by its relative line number in the file.

You must enclose the string in quotes. For example:

FIND "Fool's Paradise" BOOK1.TXT BOOK2.TXT

displays all lines from BOOK1.TXT and BOOK2.TXT (in that order) that contain the string "Fool's Paradise". The command:

DIR B: | FIND /V "DAT"

causes MS-DOS to display all names of the files on the disk in drive B which do not contain the string DAT. Type double quotes around a string that already has quotes in it.

Note: FIND distinguishes between uppercase and lowercase letters while searching for the specific string.

FIND, *cont.*

When MS-DOS detects an error, FIND responds with one of the following error messages:

Incorrect DOS version

FIND runs only on versions of MS-DOS that are 2.0 or higher.

FIND: Invalid number of parameters

You did not specify a string when issuing the FIND command.

FIND: Syntax error

You typed an illegal string when issuing the FIND command.

FIND: File not found <filespec>

The file you have specified does not exist or FIND cannot find it.

FIND: Read error in <filespec>

An error occurred when FIND tried to read the file specified in the command.

FIND: Invalid parameter <option name>

You specified an option that does not exist.

NAME: FORMAT**TYPE:** External**PURPOSE:**

Formats the disk in the specified drive to accept MS-DOS files.

SYNTAX:

FORMAT [d:][/V] $\left\{ \begin{array}{l} /8 \\ /9 \end{array} \right\}$ [/1][/0][/C][/D][/S]

NOTES:

This command initializes the directory and file allocation tables. If you do not specify a drive, MS-DOS formats the disk in the default drive.

You may specify the switches in any order. Refer to Section 2 under "The FORMAT Command" for descriptions of each switch.

NAME: MKDIR**TYPE:** Internal**SYNONYM:**

MD

PURPOSE:

Makes a new directory.

SYNTAX:

MKDIR <pathname>

NOTES:

You can use this command to create a hierarchical directory structure. When you are in your root directory, you can create subdirectories by using the MKDIR command. The command:

MKDIR\USERS

creates a subdirectory \USERS in your root directory. To create a directory named JOE under \USERS, type:

MKDIR\USERS\JOE

NAME: MODE**TYPE:** External**PURPOSE:**

Sets the operation parameters for printer output, screen display, and the RS-232-C Interface Module. This command can also redirect the standard printer output from a Printer Module to an RS-232-C Module.

SYNTAX:

1. To set the printer mode:

```
MODE [ { LPTn: } ] [ { 80 } ] [ [ { 6 } ] [ , P ] ]
      [ { PRN: } ] [ { 132 } ] [ , [ { 8 } ] ]
```

2. To set the screen mode:

```
MODE [ { 40 } ] [ { 80 } ] [ { ,BW } ] [ { ,CH } ] [ { ,NM } ]
      [ { ,CO } ] [ { ,GR } ] [ { ,EM } ]
      [ { ,C16 } ]
```

3. To set the communication parameters for an RS-232-C Module:

```
MODE COMn:baud[, [parity][, [databits][, [stopbits][, P]]]]
```

4. To redirect Printer Module output to an RS-232-C Module:

```
MODE LPTn: = COMn
```

NOTES:

Because the MODE command is an external command, you cannot use it unless the MODE.COM file exists on your disk. The MODE command displays different verification messages depending on which form of the command you enter.

1. To set the printer mode:

MS-DOS can support up to three printers in a system. Corresponding to these three printers, the n variable in the LPTn: parameter can be 1, 2, or 3. PRN: refers to the first printer in the system. Therefore, PRN: and LPT1: are interchangeable in this case.

The second parameter in the MODE command for the printer determines the maximum number of characters in each printed line. If you are using 8 1/2-inch wide paper or your printer prints only 80 characters per line, then you should set this parameter to 80. Otherwise, you can

MODE, cont.

obtain the maximum flexibility by specifying 132 as the second parameter. Eighty characters per line is the default line length.

The third parameter in the MODE command for the printer specifies the line spacing in lines per inch. The default is 6 lines per inch but you can specify 8 lines per inch to fit more lines of text on each page.

2. To set the screen mode:

The first parameter in this form of the MODE command enables you to set the number of characters per line on the screen to either 40 or 80.

When using the MODE command to set the screen mode, the second parameter selects black and white, 4-color, or 16-color operation. BW selects black-and-white operation. CO (the letter O, not zero) selects 4-color operation and C16 selects 16-color operation.

The third parameter selects character mode or graphics mode operation. CH and GR represent character mode and graphics mode, respectively.

The last parameter for screen mode selection determines whether the system operates in the Mindset-Native mode (selected with NM), or in the industry-standard emulation mode (selected with EM). NM allows you to use all of the multicolor and high resolution features of the Mindset Personal Computer.

The parameters, other than the number of characters per line, may be entered in any order. Any parameters that are omitted, remain unchanged.

3. To set the communication parameters for an RS-232-C Module:

The MODE command can specify the communication parameters for the data transmitted and received through the RS-232-C Module. You should consult the documentation for the serial-interface device you are using (for example, a printer, plotter, or remote computer) to determine the correct settings for these communication parameters.

The variable *n* in the COM*n*: parameter is 1, 2, or 3, depending on whether you are specifying parameters for the first or the second RS-232-C Modules. If you have only one RS-232-C module, it is COM1.

The baud parameter sets the data transfer rate in terms of bits per second. There is no default baud rate value so you must specify one of the following: 110, 150, 300, 600, 1200, 2400, 4800, or 9600. You can also abbreviate these values by entering only the first two digits: 11, 15, 30, 60, 12, 24, 48, or 96.

Computers use parity to check received characters for possible transmission errors. You can specify the parity parameter as E for even parity, O for odd parity, or N for no parity. Even parity is the default setting if you omit this optional specification.

Some serial-interface devices use 7 data bits to represent a character while others use 8. You should specify the databits parameter as either 7 or 8, depending on the requirements of the device with which you are communicating. If you omit this parameter, then it defaults to 7 data bits per character.

Devices operating at 110 baud usually require two stop bits to signal the end of each character, and devices operating at higher data rates usually require only one stop bit. For this reason, the value of the stopbits parameter defaults to 2 when you specify 110-baud operation and to 1 when you specify any other baud rate. Usually, you do not need to specify this parameter.

The optional letter P specifies that you are using a serial-interface printer (or plotter) as opposed to linking the Mindset Personal Computer with another computer. Including the P causes the system to continue trying to access the printer when it does not detect a response. This method gives a slow printer sufficient time to respond to the data it receives from the computer.

If the printer does not respond after a few seconds, you can use the BREAK key to halt the retry procedure. If you do not include the P specification, the system tries once and then quits if it does not detect a response from the serial-interface device.

The parameters following the baud rate are optional. You can simply omit them and accept the default values. If you want to omit an optional parameter and still specify another which occurs later in the command, you can skip it by entering just the comma which normally follows the parameter. For example,

MODE COM1:30,,8

MODE, *cont.*

specifies 300-baud operation, the default parity (even), 8 data bits per character, and the remaining two defaults of one stop bit and no P (no retries).

4. To redirect Printer Module output to an RS-232-C Module:

This form of the **MODE** command enables you to use an RS-232-C Module in place of a Printer Interface Module for output. After you enter this command, you can use an RS-232-C Module with a serial-interface printer to produce the output from the PRT SCN (print screen) key and the **PRINT** statement in BASIC.

The variable *n* in the **LPTn** parameter represents the number of the parallel printer which you want to replace with a serial printer. You can enter 1, 2, or 3 for the *n* in **LPTn**. The variable *n* in the **COMn** parameter represents the number of the RS-232-C Module which is to replace the Printer Interface Module. You can enter 1 or 2 for the *n* in **COMn**.

For example, if your system has only an RS-232-C Module for printer output, enter the following command:

MODE LPT1: = COM1

5. To clear redirection:

Redirection is cleared by issuing a new setting for the redirection printer, for example:

MODE LPTn: $\left[\left\{ \begin{array}{c} 80 \\ 132 \end{array} \right\} \right] \left[\left[\left\{ \begin{array}{c} 6 \\ 8 \end{array} \right\} \right] \left[, P \right] \right]$

or,

MODE LPTn:

6. To determine status:

Display the current printer screen or communications port by simply typing:

MODE

NAME: MORE**TYPE:** External**PURPOSE:**

Sends output to the console one screen at a time.

SYNTAX:

MORE

NOTES:

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

Press the RETURN key to display another screen of information. Continue this process until MS-DOS displays all the input data.

The MORE command is useful for viewing a long file one screen at a time. If you type:

TYPE MYFILES.COM | MORE

MS-DOS displays the file MYFILES.COM (on the default drive) one screen at a time.

NAME: PATH**TYPE:** Internal**PURPOSE:**

Sets a command path.

SYNTAX:

PATH [<pathname>[;<pathname>]...]

NOTES:

This command enables you to tell MS-DOS which directories to search for external commands after it searches your working directory. The default value is no path.

To tell MS-DOS to search your \BIN\USERS\JOE directory for external commands, type:

```
PATH \BIN\USERS\JOE 
```

MS-DOS searches the \BIN\USERS\JOE directory for external commands until you set another path or shut down MS-DOS.

You can tell MS-DOS to search more than one path by specifying several pathnames separated by semicolons. For example,

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

tells MS-DOS to search the directories specified by the above pathnames to find external commands. MS-DOS searches the pathnames in the order specified in the PATH command.

The command PATH with no options displays the current path on the screen. If you specify **PATH ;**, MS-DOS sets the NUL path, meaning that it will search only the working directory for external commands.

NAME: PRINT**TYPE:** External**PURPOSE:**

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

SYNTAX:

PRINT [[d:][filespec][/T][/C][/S]]...

NOTES:

You can use the PRINT command only if a line printer is attached to your computer. MS-DOS provides the following switches with the PRINT command:

/T TERMINATE: Deletes all files in the print queue (files waiting to be printed). MS-DOS prints a message to this effect.

/C CANCEL: Turns on cancel mode. MS-DOS cancels the preceding filespec and all following filespecs in the print queue.

/P PRINT: Turns on print mode. MS-DOS adds the preceding filespec and all following filespecs to the print queue until you issue a /C switch.

PRINT with no options displays the contents of the print queue on your screen without affecting the queue. PRINT with options performs the following tasks:

PRINT /T: empties the print queue.

PRINT /T *.ASM: empties the print queue and queues all .ASM files on the default drive.

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.

If MS-DOS detects an error, PRINT displays one of the following error messages:

Name of list device [PRN:]

PRINT, *cont.*

This prompt appears when PRINT is run the first time. You may specify any current device and that device then becomes the PRINT output device. As indicated in the brackets, simply pressing RETURN causes MS-DOS to use the PRN device.

List output is not assigned to a device

MS-DOS displays this message if the "Name of list device" specified to the above prompt is invalid. Subsequent attempts will return the same message until you specify a valid device.

PRINT queue is full

There is room for 10 files in the queue. If you attempt to put more than 10 files in the queue, this message appears on the screen.

PRINT queue is empty

There are no files in the print queue.

No files match d:XXXXXXXXX.XXX

No files match the filespec which you used to specify files to add to the print queue. Note that if there are no files in the queue to match the canceled filespec, no error message appears.

Drive not ready

If this message occurs when PRINT attempts a disk access, PRINT keeps trying until the drive is ready. Any other error causes MS-DOS to cancel the printing of the current file. In this case, MS-DOS sends an error message to your printer.

All files canceled

If you use the /T (TERMINATE) switch, MS-DOS sends the message "All files canceled by operator" to your printer. If you cancel the current file being printed with a /C, MS-DOS prints the message "File canceled by operator".

NAME: PROMPT**TYPE:** Internal**PURPOSE:**

Changes the MS-DOS command prompt.

SYNTAX:

PROMPT [<prompt-text>]

NOTES:

This command enables you to change the MS-DOS system prompt (for example, A>). If you do not include any prompt text, MS-DOS sets the prompt to the default prompt, which is the default drive designation. You can set the prompt to a special prompt, such as the current time, by using the characters indicated below.

You must precede the characters with a dollar sign (\$) in the prompt command. The left column indicates the character you specify to obtain the prompt in the right column.

Character	Prompt
\$	The \$ character
t	The current time
d	The current date
p	The current directory of the default drive
v	The version number
n	The default drive
g	The > character
	The < character
b	The character
_	A CR LF sequence
s	A space (leading only)
h	A backspace
e	ASCII code X1B (escape)

For example:

```
PROMPT $n $g
```

sets the normal MS-DOS prompt (d>), where d is the default drive.

```
PROMPT Time = $t $__Date = $d
```

PROMPT, *cont.*

sets a two-line prompt which prints:

Time = (current time)
Date = (current date)

If your keyboard has an ANSI escape sequence driver, then you can use escape sequences in your prompts. For example:

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

Sets the prompts in inverse video mode and returns to video mode for other text.

NAME: RECOVER**TYPE:** External**PURPOSE:**

Recovers a file or an entire disk containing bad sectors.

SYNTAX:

RECOVER <filespec | d:>

NOTES:

If a sector on a disk is bad, you can recover either the file containing that sector (without the bad sector) or the entire disk (if the bad sector was in the directory).

To recover a particular file, type:

RECOVER <filespec>

MS-DOS reads the file sector by sector and skips the bad sector(s). When MS-DOS finds the bad sector(s), it marks the sector(s) and no longer allocates your data to that sector.

To recover a disk, type:

RECOVER <d:>

where d: is the letter of the drive containing the disk you want to recover.

If there is not enough room in the root directory, RECOVER prints a message and stores information about the extra files in the File Allocation Table. You can run RECOVER again to regain these files when there is more room in the root directory.

When recovering a disk, the recovered files will be renamed "FILE nnnn". This procedure is especially important on system diskettes where the files IO SYS and MS DOS.SYS have been renamed. Renaming the files makes the recovered diskette unbootable.

NAME: REM (REMARK)**TYPE:** Internal**PURPOSE:**

Displays remarks which are on the same line as the REM command in a batch file during execution of that batch file.

SYNTAX:

REM [comment]

NOTES:

The only separators allowed in the comment are the space, tab, and comma.

For example:

- 1: **REM This file checks new disks**
- 2: **REM It is named NEWDISK. BAT**
- 3: **PAUSE Insert new disk in drive B:**
- 4: **FORMAT B:/S**
- 5: **DIR B:**
- 6: **CHKDSK B:**

NAME: TYPE**TYPE:** Internal**PURPOSE:**

Displays the contents of the file on the screen.

SYNTAX:

Type [d:] [pathname] <filespec>

NOTES:

Use this command to examine a file without modifying it. (Use DIR to find the name of a file and EDLIN to alter the contents of a file.) The only formatting that TYPE performs is the expansion of tabs to spaces consistent with tab stops every eighth column. Note that a display of binary files causes MS-DOS to send control characters (such as CTRL-Z) to your computer, including bells, form feeds, and escape sequences.

NAME: VER

TYPE: Internal

PURPOSE:

Prints the MS-DOS version number.

SYNTAX:

VER

NOTES:

If you want to know what version of MS-DOS you are using, type **VER**. MS-DOS responds by displaying the version number on your screen.

NAME: VERIFY**TYPE:** Internal**PURPOSE:**

Turns the verify switch on or off when writing to disk.

SYNTAX:

VERIFY [ON|OFF]

NOTES:

This command has the same purpose as the /V switch in the COPY command. If you want to verify that all files are written correctly to disk, you can use the VERIFY command to tell MS-DOS to verify that your files are intact (for example, no bad sectors). MS-DOS performs a VERIFY each time you write data to a disk. You receive an error message only if MS-DOS is unable to successfully write your data to disk.

VERIFY ON remains in effect until you change it in a program (by a SET VERIFY system call), or until you issue a VERIFY OFF command to MS-DOS.

If you want to know the current setting of VERIFY, type **VERIFY** with no options.

NAME: VOL (VOLUME)

TYPE: Internal

PURPOSE:

Displays the disk volume label, if it exists.

SYNTAX:

VOL [d:]

NOTES:

This command prints the volume label of the disk in drive d:. If you do not specify a drive, MS-DOS prints the volume label of the disk in the default drive.

If the disk does not have a volume label, VOL displays:

Volume in drive x has no label

Batch Processing Commands

The following commands are called batch processing commands. They can add flexibility and power to your batch programs. The commands discussed are ECHO, FOR, GOTO, IF, PAUSE, and SHIFT.

If you are not writing batch programs, you do not need to read this section.

NAME: ECHO**TYPE:** Internal**PURPOSE:**

Turns the batch echo feature on and off.

SYNTAX:

ECHO [ON|OFF| message]

NOTES:

Normally, commands in a batch file are displayed (“echoed”) on the console when they are seen by the command processor. ECHO OFF turns off this feature. ECHO ON turns the echo back on.

If ON or OFF are not specified, ECHO displays the current setting.

NAME: FOR**TYPE:** Internal**PURPOSE:**

Command extension used in batch and interactive file processing.

SYNTAX:

FOR %%<c> IN <set> DO <command>—for batch processing

FOR % <c> IN <set> DO <command>—for interactive processing

NOTES:

<c> can be any character except 0,1,2,3,..9 to avoid confusion with the %0-%9 batch parameters.

<set> is (#<item>*#)

MS-DOS sets the %%c variable sequentially to each member of set, and then evaluates command. If a member of set is an expression involving * and/or ?, then FOR sets the variable to each matching pattern from disk. In this case, only one such item may be in the set, and FOR ignores any item besides the first.

For example:

```
FOR %%f IN ( *.ASM ) DO MASM %%f;  
FOR %%f IN (FOO BAR BLECH) DO REM %%f
```

The '%%' is needed so that after batch parameter (%0-%9) processing is done, there is one '%' left. If only '%f' were there, the batch parameter processor would see the '%', look at 'f', decide that '%f' was an error (bad parameter reference), and throw out the '%f', so that the command FOR would never see it. If the FOR is not in a batch file, then only one '%' should be used.

NAME: GOTO**TYPE:** Internal**PURPOSE:**

Command extension used in batch file processing.

SYNTAX:

GOTO <label>

NOTES:

GOTO causes MS-DOS to take commands from the batch file beginning with the line after the label definition. If you have not defined a label, the current batch file is terminated.

For example:

```
:foo  
REM looping...  
GOTO foo
```

produces an infinite sequence of messages:

```
REM looping....
```

Starting a line in a batch file with ':' causes batch processing to ignore the line. The characters following GOTO define a label, but you must also use this procedure to put in comment lines. Labels cannot contain the "." character.

NAME: IF**TYPE:** Internal**PURPOSE:**

Command extension used in batch file processing.

SYNTAX:

IF <condition> <command>

NOTES:

The condition is one of the following:

ERRORLEVEL <number>: True if and only if the previous program executed by COMMAND had an exit code of number or higher.

<string1> = = <string2>: True if and only if string1 and string2 are identical after parameter substitution. Strings may not include embedded separators.

EXIST <filespec>: True if and only if filespec exists.

NOT <condition>: True if and only if condition is false.

The IF statement allows conditional execution of commands. When the condition is true, then MS-DOS executes the command. Otherwise, MS-DOS ignores the command.

For example:

```
IF NOT EXIST \TMP\FOO ECHO Can't find file
```

```
IF NOT ERRORLEVEL 3 LINK $1,,;
```

NAME: PAUSE**TYPE:** Internal**PURPOSE:**

Suspends execution of the batch file.

SYNTAX:

PAUSE [comment]

NOTES:

During the execution of a batch file, you may need to change disks or perform some other action. PAUSE suspends execution until you press any key, except the CTRL-C sequence.

When the command processor encounters PAUSE, it prints:

Strike a key when ready...

If you press CTRL-C, MS-DOS displays the following prompt:

Abort batch job (Y/N)?

If you type **Y** in response to this prompt, MS-DOS aborts execution of the remainder of the batch command file and returns control to the operating system command level. Therefore, you can use PAUSE to break a batch file into pieces, allowing you to end the batch command file at an intermediate point.

The comment is optional; you may enter it on the same line as PAUSE. You may also want to prompt the user of the batch file with some meaningful message when the batch file pauses. For example, you may want to change disks in one of the drives. You may supply an optional prompt message in such cases. MS-DOS displays the comment prompt before the "Strike a key when ready..." message.

NAME: SHIFT**TYPE:** Internal**PURPOSE:**

Provides more than 10 replaceable parameters in batch file processing.

SYNTAX:

SHIFT

NOTES:

Usually, MS-DOS limits command files to handling 10 parameters, %0 through %9. To allow access to more than ten parameters, use SHIFT to change the command line parameters. For example:

```
if    %0 = "foo"  
      %1 = "bar"  
      %2 = "name"  
      %3...%9 are empty
```

then a SHIFT will cause:

```
%0 = "bar"  
%1 = "name"  
%2...%9 are empty
```

If more than 10 parameters are given on a command line, MS-DOS shifts those that appear after the 10th (%9) one at a time into %9 by successive shifts.



MS-DOS Editing and Function Keys

Special MS-DOS Editing Keys

The special editing keys deserve particular emphasis because they depart from the way in which most operating systems handle command input. You do not have to type the same sequences of keys repeatedly, because the last command line is automatically placed in a special storage area called a template.

By using the template and the special editing keys, you can take advantage of the following MS-DOS features:

1. You can instantly repeat the preceding command line by pressing two keys.
2. If you make a mistake in the command line, you can edit it and retry without having to retype the entire command line.
3. You can edit and execute a command line that is similar to a preceding command line with a minimum of typing by pressing a special editing key.

Figure 6-1 shows the relationship between the command line and the template.

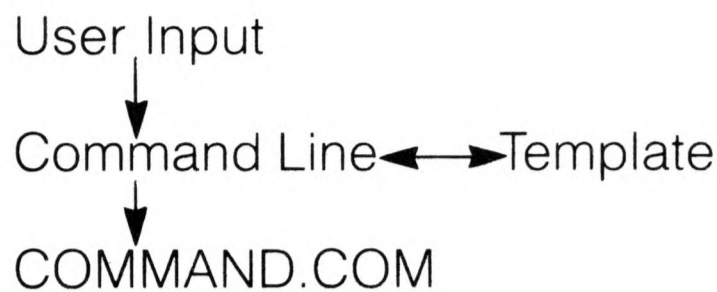








Figure 6-1: Command line and template

As seen in Figure 6-1, you type a command to MS-DOS on the command line. When you press the RETURN key, the command is automatically sent to the command processor (COMMAND.COM) for execution. At the same time, MS-DOS sends a copy of this command to the template. You can now recall the command or modify it with MS-DOS special editing keys.

Table 6-1 contains a complete list of the special editing keys. Section 7, "The Line Editor (EDLIN)", describes each of these keys more fully. You use these commands in EDLIN to edit your text files.

Table 6-1: Special Editing Functions

Key	Editing Function
	Copies one character from the template to the command line.
	Copies characters up to the character specified in the template and puts these characters on the command line.
	Copies all remaining characters in the template to the command line.
	Skips over (does not copy) a character in the template.
	Skips over (does not copy) the characters in the template up to the character specified.
	voids the current input; leaves the template unchanged.

INS	Enters and exits INS mode.
F5	Makes the new line the new template.
CTRL-Z	Puts a CTRL-Z (1AH) end-of-file character in the new template.
◀	Deletes one character from the new line.

For example, if you type the following command:

DIR PROG.COM **RETURN**

MS-DOS displays information about the file PROG.COM on your screen. MS-DOS also saves the command line in the template. To repeat the command, just press two keys: F3 and RETURN.

MS-DOS executes the repeated command on the screen after you press RETURN, as shown below:

F3 **DIR PROG.COM** **RETURN**

Notice that pressing the F3 key causes MS-DOS to copy the contents of the template to the command line; pressing RETURN causes MS-DOS to send the command line to the command processor for execution.

If you want to display information about a file named PROG.ASM, you can use the contents of the template and type:

F2 **C** **RETURN**

Typing F2 C copies all characters from the template to the command line, up to but not including C. MS-DOS displays:

DIR PROG.

Note that the underline is your cursor. Now type:

ASM **RETURN**

The result is:

DIR PROG.ASM

The command line DIR PROG.ASM is now in the template and ready to be sent to the command processor for execution. To do this, press RETURN.

Now assume that you want to execute the following command:

TYPE PROG.ASM

To do this, type:

TYPE [INS] [F3] [RETURN]

Notice that when you are typing, MS-DOS simultaneously enters the characters directly into the command line and overwrites corresponding characters in the template. Pressing the INS key turns off this automatic replacement feature. Thus, the characters TYPE replace the characters DIR in the template. To insert a space between TYPE and PROG.ASM, press INS and then the space bar. Finally, to copy the rest of the template to the command line, press F3 and then RETURN. MS-DOS processes the command TYPE PROG.ASM and the template becomes TYPE PROG.ASM.

If you misspell TYPE as BYTE, a command error occurs. Still, instead of throwing away the whole command, you can save the misspelled line before you press RETURN by creating a new template with the F5 key:

BYTE PROG.ASM [F5] [RETURN]

You can then edit this erroneous command by typing:

T [▶] **P** [F3] [RETURN]

The ▶ key copies a single character from the template to the command line. The resulting command line is then the command that you want:

TYPE PROG.ASM

As an alternative, you can use the same template containing BYTE PROG.ASM and then use the DEL and INS keys to achieve the same result:

[DEL] [DEL] [▶] [INS] **YP** [F3] [RETURN]

To illustrate how these editing keys affect the command line as you type, examine the keys typed on the left; their effect on the command line is described on the right:

DEL	—	Discards 1st template character
DEL	—	Discards 2nd template character
▶	T_	Copies 3rd template character
INS YP	TYP_	Inserts two characters
F3	TYPE PROG.ASM	Copies remainder of template

Notice that DEL does not affect the command line. It affects the template by deleting the first character. Similarly, F4 deletes characters in the template, up to but not including a given character.

These special editing keys can add to your effectiveness at the keyboard. The next section describes control sequences that can also help when you are typing commands.

Control Sequences

A control sequence is a function that affects the command line. You have already learned about CTRL-C and CTRL-S. This section describes the other control sequences of MS-DOS.

Remember that when you type a control sequence, such as CTRL-C, you must hold down the control key and then press the C key.

Table 6-2: Control Sequences

Control Sequence	Function
CTRL-N	Toggles echoing of output to line printer.
CTRL-C	Aborts current command.
CTRL-H	Backspaces to remove last character from command line, and erases character from the screen.

Control Sequence	Function
CTRL-J	Inserts physical end-of-line, but does not empty command line. Use the LINE FEED key to extend the current logical line beyond the physical limits of the screen.
CTRL-P	Toggles screen output to line printer.
CTRL-S	Suspends output display on screen. Press any key to resume.
CTRL-X	Cancels the current line, empties the command line, and then displays a back slash (\) and places the cursor on the next line. This control character does not affect the template used by the special editing commands.

The Line Editor (EDLIN)

Line Editor

Section 7

The Line Editor (EDLIN)

Introduction

In this section you learn how to use EDLIN, the line editor program. You can use EDLIN to create, change, and display files, whether they are program source files or text files.

You can use EDLIN to:

- Create and save new source files.
- Update existing files and save both the updated and original files.
- Delete, edit, insert, and display lines within files.
- Search for, delete, or replace text within one or more lines of a file.

EDLIN stores text in lines, and each line may contain up to 253 characters. EDLIN generates and displays line numbers during the editing process, but the numbers are not actually present in the saved file.

When you insert lines, all line numbers following the inserted text advance automatically by the number of lines you are inserting. When you delete lines in a file, all line numbers following the deleted text

decrease automatically by the number of lines deleted. As a result, you always see the lines numbered consecutively in your file.

Starting EDLIN

To start EDLIN, type:

```
EDLIN <d:> <filespec> [\B] RETURN
```

If you are creating a new file, the filespec should be the name of the file you wish to create. If EDLIN does not find this file on a drive, EDLIN creates a new file with the name you specify and displays the following message and prompt:

```
New file  
*  
—
```

Notice that the prompt for EDLIN is an asterisk (*).

You can now type lines of text into your new file. To begin entering text, you must enter an I (Insert) command to insert lines. The I command is discussed later in this section.

If you want to edit an existing file, filespec should be the name of the file you want to edit. When EDLIN finds the file you specify on the designated or default drive, it loads the file into memory. If there is enough room in memory for you to load the entire file, EDLIN displays the following message on your screen:

```
End of input file  
*
```

You can then edit the file using EDLIN editing commands.

If the file is too large to be loaded into memory, EDLIN loads lines until memory is 75% full, and then displays the * prompt. You can then edit the portion of the file that is in memory.

To edit the remainder of the file, you must save some of the edited lines on disk to free memory; then EDLIN can load the unedited lines from disk into memory. Refer to the Write and Append commands in this section for the procedure.

When you complete the editing session, you can save the original and the updated (new) files by using the End command. This section discusses the End command under "EDLIN Commands". EDLIN renames the original file with an extension of .BAK, and the new file has the filename and extension you specify in the EDLIN command. The original .BAK file is not erased until the end of the editing session, or until disk space is needed by the editor (EDLIN).

Do not try to edit a file with a filename extension of .BAK because EDLIN assumes that any .BAK file is a backup file. If you find it necessary to edit such a file, rename the file with another extension (using the MS-DOS RENAME command discussed in Section 5), then start EDLIN and specify the new filespec.



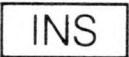


Special Editing Keys

You can use the special editing keys and template discussed in Section 6 to edit your text files. This section discusses these keys in detail.

Table 7-1 summarizes the functions of the special editing keys. Detailed descriptions of the special editing key functions follow the table.

Table 7-1: Special Editing Keys

Function	Key	Description
Copy one character	▶	Copies one character from the template to the new line.
Copy up to character	F2	Copies all characters from the template to the new line, up to the character specified.
Copy template	F3	Copies all remaining characters in the template to the screen.
Skip one character	DEL	Skips over (does not copy) a character.

Function	Key	Description
Skip up to character		Skips over (does not copy) the characters in the template, up to the character specified.
Quit input		voids the current input; leaves the template unchanged.
Insert mode		Enters/exits character insert mode.
New template		Makes the new line the new template.
Back up one character		Backspaces over and deletes one character in the new line.

KEY: ▶**PURPOSE:**

Copies one character from the template to the new line.

NOTES:

Pressing the ▶ key copies one character from the template to the new line. When you press the ▶ key, EDLIN enables you to insert one character in the new line and then turns off the insert mode automatically.

EXAMPLE:

Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session, EDLIN positions the cursor at the beginning of the line. Pressing the ▶ key copies the first character (T) to the second of the two lines displayed:

```
1:*This is a sample file.
▶ 1:*T_
```

Each time you press the ▶ key, one more character appears from the template:

```
▶ 1:*Th_
▶ 1:*Thi_
▶ 1:*This_
```

KEY: F2**PURPOSE:**

Copies multiple characters up to a given character.

NOTES:

Pressing the F2 key copies all characters up to a given character from the template to the new line. The given character is the next character typed after F2; EDLIN does not copy or display this character on the screen when you type it. When you type the character after pressing the F2 key, EDLIN copies text from the template up to, but not including, the first occurrence of the given character. If the template does not contain the specified character, MS-DOS does not copy any text from the template. Pressing F2 also automatically turns off insert mode.

EXAMPLE:

Assume that the screen shows:

```
1:*This is a sample file.  
1:*__
```

At the beginning of the editing session, EDLIN positions the cursor at the beginning of the line. Pressing the F2 key copies all characters up to the character specified immediately after the F2 key.

```
1:*This is a sample file.  
[F2] p 1:*This is a sam__
```

KEY: F3**PURPOSE:**

Copies the template to the new line.

NOTES:

Pressing the F3 key copies all remaining characters from the template to the new line. Regardless of the cursor position when you press the F3 key, the remaining portion of the template line appears, and EDLIN positions the cursor after the last character on the line.

EXAMPLE:

Assume that the screen shows:

```
1:*This is a sample file.  
1:*_
```

At the beginning of the editing session, EDLIN positions the cursor at the beginning of the line. Pressing the F3 key copies all characters from the template (shown in the upper line displayed) to the line with the cursor (the lower line displayed):


```
1:*This is a sample file. (template)  
[F3] 1:*This is a sample file.__(command line)
```

F3 also turns off the insert mode automatically.

KEY: DEL**PURPOSE:**

Skips over one character in the template.

NOTES:

Pressing the DEL key skips over one character in the template. Each time you press the DEL key, one character is not copied from the template. The action of the DEL key is similar to the  key, except that DEL skips a character in the template instead of copying it to the new line.

EXAMPLE:

Assume that the screen shows:

```
1:*This is a sample file.
1:*  _
```

At the beginning of the editing session, EDLIN positions the cursor at the beginning of the line. Pressing the DEL key skips over the first character (T).

```
1:*This is a sample file.
[DEL] 1:*  _
```

Pressing DEL affects only the template; it does not change the cursor position. To see how much of the template line remains, press the F3 key to display the remaining text and move the cursor beyond the last character of the line.

```
1:*This is a sample file.
[DEL] 1:*  _
[F3] 1:*his is a sample file.  _
```

KEY: F4**PURPOSE:**

Skips multiple characters in the template up to the specified character.

NOTES:

Pressing the F4 key skips over all characters up to a given character in the template. You specify the given character by typing it after pressing F4. EDLIN responds by deleting text from the template beginning with the first character and ending just before the first occurrence of the specified character. The action of F4 is not apparent until you use one of the copy commands to display the remaining template text.

If the template does not contain the specified character, F4 has no effect and the template remains unchanged. The action of the F4 key is similar to the F2 key, except that F4 skips over characters in the template instead of copying them to the new line.

EXAMPLE:

Assume that the screen shows:

```
1:*This is a sample file.
1:* _
```

At the beginning of the editing session, EDLIN positions the cursor at the beginning of the line. Pressing the F4 key skips over all the characters in the template up to, but not including, the character pressed after the F4 key:

```
1:*This is a sample file.
[F4] p 1:* _
```

The cursor position does not change. To see how much of the template line remains, press the F3 key to copy the template to the command line. After the F3 command displays the template text, it moves the cursor beyond the last character of the line:

```
1:*This is a sample file.
[F4] p 1:* _
[F3] 1:*ple file. _
```

KEY: ▼**PURPOSE:**

Quits input and empties the new line.

NOTES:

Pressing the ▼ key empties the new line, but leaves the template unchanged. ▼ also prints a backslash (\), carriage return, and line feed, and turns insert mode off. EDLIN positions the cursor at the beginning of the line. Pressing the F3 key copies the template to the new line and the command line appears as it was before you pressed ▼.

EXAMPLE:

Assume that the screen shows:

```
1:*This is a sample file.
1:*  _
```

At the beginning of the editing session, EDLIN positions the cursor at the beginning of the line. Assume that you want to replace the line with "Sample File":

```
1:*This is a sample file.
1:*Sample File_
```

To cancel the line you just entered (Sample File), and to keep "This is a sample file.", press ▼ before you press RETURN. Notice that a backslash appears on the "Sample File" line to tell you EDLIN has canceled it.

```
1:*This is a sample file.
▼ 1:*Sample File\
1:  _
```

Press RETURN to keep the original line, or to perform any other editing functions. If you press F3, EDLIN copies the original template to the command line:

```
[F3] 1: This is a sample file. _
```

KEY: INS**PURPOSE:**

Enters or exits from character insertion mode.

NOTES:

Pressing the INS key causes EDLIN to enter or exit from character insertion mode. The character insertion mode is different from the EDLIN insert command (I) which enables you to insert new lines. The current cursor position in the template remains unchanged. The cursor moves as you insert each character. However, when you have finished inserting characters, EDLIN positions the cursor at the same character as it was before the insertion began. Thus, you are inserting characters in front of the character at which the cursor points.

EXAMPLE:

Assume that the screen shows:

```
1:*This is a sample file.
1:* _
```

At the beginning of the editing session, EDLIN positions the cursor at the beginning of the line. Assume that you press the F2 and f keys:

```
1:*This is a sample file.
[F2] f 1:*This is a sample _
```

Now press the INS key and insert the characters "edit" and a space:

```
1:*This is a sample file.
[F2] f 1:*This is a sample _
[INS] edit 1:*This is a sample edit _
```

If you now press the F3 key, EDLIN copies the rest of the template to the line:

```
1:*This is a sample edit _
[F3] 1:*This is a sample edit file. _
```

If you press the RETURN key instead of F3, EDLIN deletes the remainder of the template, and the new line ends at the end of the insert:

```
[INS] edit [RETURN] 1:*This is a sample edit _
```

To exit from insert mode, simply press the INS key again.

KEY: F5**PURPOSE:**

Creates a new template.

NOTES:

Pressing the F5 key copies the current new line to the template and discards the old template. F5 displays an @ (at-sign character) on the command line and then places the cursor on the next line. F5 exits from insert mode but does not execute the new line.

F5 performs the same function as the ▼ key, except that F5 overlays the template text and displays an @ (at-sign character) instead of a \ (backslash).

EXAMPLE:

Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session, EDLIN positions the cursor at the beginning of the line. Assume that you press F2 m, INS lary, INS tax, and then F3:

```
1:*This is a sample file.
[F2] m 1:*This is a sa__
[INS] lary 1:*This is a salary__
[INS] tax 1:*This is a salary tax__
[F3] 1:*This is a salary tax file.__
```

At this point, assume that you want this line to be the new template, so you press the F5 key:

```
[F5] 1:*This is a salary tax file.@
```

The @ indicates that this new line is now the new template. You can perform additional editing using the new template.

KEY: ◀

PURPOSE:

Backspaces over and deletes one character in the new line.

Command Information

EDLIN commands perform editing functions on lines of text. The following list contains information you should read before you use EDLIN commands.

1. Pathnames are acceptable as options for commands. For example, typing **EDLIN \BIN\USERS\JOE\TEXT.TXT** enables you to edit the TEXT.TXT file in the subdirectory JOE.
2. You can reference line numbers relative to the current line (the line with the asterisk). Use a minus sign with a number to indicate lines before the current line. Use a plus sign with a number to indicate lines after the current line.

Example:

```
-10, +10L RETURN
```

This command lists 10 lines before the current line, the current line, and 10 lines after the current line.

3. You may issue multiple commands on one command line. When you issue a command to edit a single line using a line number (<line>), a semicolon must separate commands on the line. Otherwise, one command may follow another without any special separators. In the case of a Search or Replace command, you may end the string with a CTRL-Z instead of a RETURN.

The following sample command line displays line 15 and then displays lines 10 through 20 on the screen.

```
15L; -5, +5L RETURN
```

The command line in the next example searches for "This string" and then displays 5 lines before and 5 lines after the line containing the matched string. If the search fails, then the displayed lines are those line numbers relative to the current line.

```
S This string CTRL-Z -5, +5L RETURN
```

-
4. You can type EDLIN commands with or without a space between the line number and command. For example, to delete line 6, the command 6D is the same as 6 D.
 5. You can insert a control character sequence (such as CTRL-C) into text by using the quote character CTRL-V before it while in insert mode. CTRL-V tells MS-DOS to recognize the next capital letter typed as a control character. It is also possible to use a control character in any of the string arguments of Search or Replace by using the special quote character. For example:

SCTRL-VZ
will find the first occurrence
of CTRL-Z in a file

RCTRL-VZCTRL-Zfoo
will replace all occurrences
of CTRL-Z in a file by "foo"

SCTRL-VCCTRL-Zbar
will replace all occurrences
of CTRL-C by "bar"

You can insert CTRL-V into the text by typing CTRL-V V.

6. The CTRL-Z character ordinarily tells EDLIN, "This is the end of the file." If you have CTRL-Z characters elsewhere in your file, you must tell EDLIN that these control characters do not mean end-of-file. Use the /B switch to tell EDLIN to ignore any CTRL-Z characters in the file.

The following table summarizes the EDLIN commands. "EDLIN Commands", later in this section, describes each command in detail.

Table 7-2: EDLIN Commands

Command	Purpose
<line>	Edits line number
A	Appends lines
C	Copies lines
D	Deletes lines
E	Ends editing
I	Inserts lines

Command	Purpose
L	Lists text
M	Moves lines
P	Pages text
Q	Quits editing
R	Replaces lines
S	Searches text
T	Transfers text
W	Writes lines

Command Options

Several EDLIN commands accept one or more options. The effect of a command option varies, depending on the command with which it is used. The following list describes each option.

<line> Line indicates a line number that you type. You must separate line numbers with a comma or a space from other line numbers, from other options, and from the command.

You may specify line in one of four ways:

- | | |
|----------------|--|
| Number | Any number less than 65534. If you specify a number larger than the largest existing line number, then line indicates the line after the last line number. |
| Period (.) | If you specify a period for line, then line indicates the current line number. The current line is the last line edited, not necessarily the last line displayed. EDLIN marks the current line with an asterisk between the line number and the first character. |
| Pound sign (#) | The pound sign indicates the line after the last line number. Specifying # for line has the same effect as specifying a number larger than the last line number. |

RETURN A carriage return entered without any of the line specifiers listed above directs EDLIN to use a default value appropriate to the command.

? The question mark option directs EDLIN to ask you if the correct string has been found in a Replace or Search command. After it asks you to verify the string, EDLIN waits for a yes or no response before continuing. You can indicate yes by typing a **Y** or by pressing RETURN. EDLIN interprets any character other than Y as a negative response.

<string> String represents text to be found, to be replaced, or to replace other text. You can use the string option only with the Search and Replace commands. You must end each string with a CTRL-Z or a RETURN (see the Replace command for details). Do not leave spaces between strings or between a string and its command letter, unless you want those spaces to be part of the string.

EDLIN Commands

This section describes EDLIN editing commands.

NAME: Append

PURPOSE:

Adds the specified number of lines from disk to the file you are editing. EDLIN adds the lines at the end of lines that are currently in memory.

SYNTAX:

[<n>]A

Where n is the optional number of lines to append to the file.

NOTES:

This command is meaningful only if the file you are editing is too large to fit into memory. When you start EDLIN, it reads lines until memory is 75% full.

To edit the remainder of the file that will not fit into memory, lines that you have already edited must be written to disk. Then you can load unedited lines from the disk file into memory with the Append command. Refer to the Write command in this section for information on how to write edited lines to disk.

Note the following items:

1. If you do not specify the number of lines to append, EDLIN appends lines to memory until available memory is 75% full. Append has no effect if available memory is already 75% full.
2. EDLIN displays the message "End of input file" only when the Append command has read the last line of the file into memory.

NAME: Copy

PURPOSE:

Copies a range of lines to a specified line number. You can copy the lines as many times as you want by using the count option.

SYNTAX:

[<line1>],[<line2>],<line3>,[<count>]C

Where line1 is the first line in the block of lines to copy and line2 is the last line. Line3 is the destination for the copied lines. Count is the number of copies you want to make.

NOTES:

If you do not specify a number in count, EDLIN copies the lines one time. If you omit line1 or line2, the default is the current line. EDLIN renumbers the file automatically after the copy operation.

The line numbers must not overlap or you will get an "Entry error" message. For example, 3,20,15C results in this error message.

EXAMPLES:

Assume that the following file exists and is ready to edit:

- 1: This is a sample file
- 2: used to show copying lines.
- 3: See what happens when you use
- 4: the Copy command
- 5: (the C command)
- 6: to copy text in your file.

You can copy this entire block of text by issuing the following command:

1,6,7C

The result is:

- 1: This is a sample file
- 2: used to show copying lines.
- 3: See what happens when you use
- 4: the Copy command
- 5: (the C command)
- 6: to copy text in your file.
- 7: This is a sample file

Copy, *cont.*

- 8: used to show copying lines.
- 9: See what happens when you use
- 10: the Copy command
- 11: (the C command)
- 12: to copy text in your file.

If you want to place the text within other text, the `line3` option should specify the line before which you want the copied text to appear. For example, assume that you want to copy lines and insert them within the following file:

- 1: This is a sample file
- 2: used to show copying lines.
- 3: See what happens when you use
- 4: the Copy command
- 5: (the C command)
- 6: to copy text in your file.
- 7: You can also use COPY
- 8: to copy lines of text
- 9: to the middle of your file.
- 10: End of sample file.

The command **3,6,10C** `RETURN` results in the following file:

- 1: This is a sample file
- 2: used to show copying lines.
- 3: See what happens when you use
- 4: the Copy command
- 5: (the C command)
- 6: to copy text in your file.
- 7: You can also use COPY
- 8: to copy lines of text
- 9: to the middle of your file.
- 10: See what happens when you use
- 11: the Copy command
- 12: (the C command)
- 13: to copy text in your file.
- 14: End of sample file.

NAME: Delete

PURPOSE:

Deletes a specified range of lines in a file.

SYNTAX:

[<line1>][,<line2>]D

Where line1 is the first line that EDLIN deletes in a block of lines ending with line2.

NOTES:

If you omit line1, that option defaults to the current line (the line with the asterisk next to the line number). If you omit line2 then EDLIN deletes only line1. When EDLIN deletes lines, the line immediately after the deleted section becomes the current line and has the same line number as line1 had before the deletion occurred.

EXAMPLES:

Assume that the following file exists and is ready to edit:

```

1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
.
.
.
25: (the D and I commands)
26: to edit the text
27:*in your file.
```

To delete multiple lines, type line1, line2D as follows:

5,24D RETURN

The result is:

```

1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit the text
7:*in your file.
```

Delete, *cont.*

To delete a single line, type:

6D

The result is:

1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: *in your file.

Next, delete a range of lines from the following file:

1: This is a sample file
2: used to show dynamic line numbers.
3: *See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.

To delete a range of lines beginning with the current line, type:

,6D

The result is:

1: This is a sample file
2: used to show dynamic line numbers.
3: *in your file.

Notice that EDLIN rennumbers the lines automatically.

NAME: Edit

PURPOSE:

Prepares a line of text for editing.

SYNTAX:

<line>

Where line is the number of the line you want to edit.

NOTES:

When you type a line number and press RETURN, EDLIN displays the line number and text. On the line below, EDLIN reprints the line number. The line is now ready for editing. You may use any of the EDLIN editing commands to edit the line. The existing text of the line serves as the template until you press the RETURN key.

If you do not type a line number before pressing RETURN, EDLIN displays the line after the current line (marked with an asterisk) for editing. If you do not need to make changes to the current line and the cursor is at the beginning or end of the line, press the RETURN key to accept the line as shown.

Warning: If you press the RETURN key while the cursor is in the middle of the line, EDLIN deletes the remainder of the line.

EXAMPLE:

Assume that the following file exists and is ready to edit and you want to insert the word "number" at the beginning of line 4:

```
1: This is a sample file.  
2: used to show  
3: the editing of line  
4: *four.
```

To edit line 4, type:

```
4 
```

EDLIN displays the contents of the line with a cursor below the line:

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

Edit, *cont.*

Now, using the INS and F3 special editing keys, type:

INS **number** 4: number__
 F3 RETURN 4: number four.
5:* __

NAME: End

PURPOSE:

Ends the editing session.

SYNTAX:

E

NOTES:

This command saves the edited file on disk, renames the original input file <filename>.BAK, and then exits from EDLIN. If you create the file during the editing session, EDLIN does not create a .BAK file.

There are no options for the E command. Therefore, you cannot tell EDLIN on which drive to save the file. You must select the drive on which you want to save the file when you start the editing session. If you do not specify a drive when you start EDLIN, EDLIN saves the file on the disk in the default drive. Of course, you can always use the MS-DOS COPY command to place the file on a different disk.

You must be sure that the disk contains enough free space for the entire file. If the disk does not contain enough free space when you end your editing session, EDLIN aborts the write operation and you may lose the entire edited file, although EDLIN might write part of the file out to the disk.

EXAMPLE:

E

After execution of the E command, MS-DOS displays the default drive prompt (for example, A>).

NAME: Insert

PURPOSE:

Inserts text immediately before the specified line.

SYNTAX:

[<line>]I

Where line is the optional number of the line before which you want to insert text.

NOTES:

If you are creating a new file, you must use the I command before you can insert text in the file. Text insertion begins at line number 1. Successive line numbers appear automatically each time you press RETURN.

EDLIN remains in insert mode until you type CTRL-Z. When the insert is complete and you exit from insert mode, the line immediately following the inserted lines becomes the current line. EDLIN increments all line numbers following the inserted section by the number of lines inserted.

If you do not specify line, the current line number is the default and EDLIN inserts the lines immediately before the current line. If line is any number larger than the last line number, or if you enter a pound sign in place of a line number, EDLIN appends the inserted lines to the end of the file. In this case, the last line you insert becomes the current line.

EXAMPLES:

Assume that the following file exists and is ready to edit:

- 1: This is a sample file
- 2: used to show dynamic line numbers.
- 3: See what happens when you use
- 4: Delete and Insert
- 5: (the D and I commands)
- 6: to edit text
- 7:*in your file.

To insert text before a specific line that is not the current line, type <line>I:

The result is:

7: __

Now, type the new text for line 7:

7: **and renumber lines**

Then to end the insertion, press CTRL-Z on the next line:

8:

Now type **L** to list the file. The result is:

1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7. and renumber lines
8:*in your file.

To insert lines immediately before the current line, type:

I

The result is:

8: __

Now, insert the following text and terminate with a CTRL-Z on the next line:

8: **so they are consecutive**

9:

To list the file and see the result, type:

L

Insert, *cont.*

The result is:

- 1: This is a sample file
- 2: used to show dynamic line numbers.
- 3: See what happens when you use
- 4: Delete and Insert
- 5: (the D and I commands)
- 6: to edit text
- 7: and renumber lines
- 8: so they are consecutive
- 9: *in your file.

To append new lines to the end of the file, type:

10I or **#I**

Either one of these commands produces the following:

10: __

Now, type the following new lines:

- 10: **The insert command can place new lines**
- 11: **in the file; there's no problem**
- 12: **because the line numbers are dynamic;**
- 13: **they'll go all the way to 65533.**

End the insertion by pressing CTRL-Z on line 14. EDLIN appends the new lines at the end of all previous lines in the file. Now type the List command, **L** :

The result is:

- 1: This is a sample file
- 2: used to show dynamic line numbers.
- 3: See what happens when you use
- 4: Delete and Insert
- 5: (the D and I commands)
- 6: to edit text
- 7: and renumber lines

-
- 8: so they are consecutive
 - 9: in your file.
 - 10: The insert command can place new lines
 - 11: in the file; there's no problem
 - 12: because the line numbers are dynamic;
 - 13: they'll go all the way to 65533.

NAME: List

PURPOSE:

Displays a range of lines, including the two lines specified.

SYNTAX:

[<line1>][,<line2>]L

Where line1 is the first line to list and line2 is the last line.

NOTES:

EDLIN provides default values if you omit one or both of the options. If you omit the line1 option as in:

,<line2> L RETURN

the display starts 11 lines before the current line and ends with the specified line. You must include the beginning comma to indicate the omitted line 1.

Note: If line2 is more than 11 lines before the current line, the display will be the same as if you omitted both options.

If you omit the <line2> option, as in:

<line1> L RETURN

EDLIN displays 23 lines, starting with the specified line.

If you omit both options, as in:

L RETURN

EDLIN displays 23 lines—the 11 lines before the current line, the current line, and the 11 lines after the current line. If there are fewer than 11 lines before the current line, EDLIN displays more than 11 lines after the current line to make a total of 23 lines.

EXAMPLES:

Assume that the following file exists and is ready to edit:

```

1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
.
.
.
15:*The current line contains an asterisk.
.
.
.
26: to edit text
27: in your file.
```

To list a range of lines without reference to the current line, type line1, line2L:

2,5L

The result is:

```

2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
```

To list a range of lines beginning with the current line, type a comma followed by line2L:

,26L

The result is:

```

15:*The current line contains an asterisk.
.
.
.
26: to edit text
```

List, *cont.*

To list a range of 23 lines centered around the current line, type only L:

L RETURN

The result is:

4: Delete and Insert
5: (the D and I commands)
. . .
13: The current line is listed in the middle.
14: The current line remains unchanged.
15:*The current line contains an asterisk.
. . .
26: to edit text.

NAME: Move

PURPOSE:

Moves a range of text to the line specified.

SYNTAX:

[<line1>],[<line2>],<line3>M

Where line1 is the first line in the block you are moving and line2 is the last line. line3 is the destination of the move operation.

NOTES:

Use the Move command to move a block of text (from the line1 to the line2) to another location in the file. EDLIN renumbers the lines according to the direction of the move. For example:

, + 25,100M

moves the text from the current line plus 25 lines to line 100. If the line numbers overlap, EDLIN displays an "Entry error" message.

To move lines 20 through 30 to line 100, type:

20,30,100M

NAME: Page

PURPOSE:

Pages through a file 23 lines at a time.

SYNTAX:

[<line1>][,<line2>]P

Where line1 is the first line to display and line2 is the last line.

NOTES:

If you omit line1, that number defaults to the current line plus one. If you omit line2, 23 lines are listed. The last line displayed becomes the current line. EDLIN marks this line with an asterisk.

NAME: Quit

PURPOSE:

Quits the editing session, does not save any editing changes, and returns to the MS-DOS operating system.

SYNTAX:

Q

NOTES:

EDLIN prompts you to make sure you don't want to save the changes.

If you type **Y** to quit the editing session, EDLIN does not save your editing changes and does not create a .BAK file. Refer to the End command in this section for information about the .BAK file.

Type **N** or any other character except Y to continue the editing session.

When started, EDLIN erases any previous copy of the file with an extension of .BAK to make room to save the new copy. Because EDLIN deletes the .BAK file when you start EDLIN, your .BAK file no longer exists when you quit EDLIN.

EXAMPLE:

```
Q
Abort edit (Y/N)?Y
A>__
```

NAME: Replace

PURPOSE:

Replaces all occurrences of a string of text in the specified range with a different string of text or blanks.

SYNTAX:

[<line1>][,<line2>][?]R<string1>[CTRL-Z<string2>]

Where line1 is the first line defining a block of text and line2 is the last line. The optional question mark causes EDLIN to stop for you to verify each replacement it makes. string1 is the string of text you want EDLIN to replace and string2 is the string you want to replace string1. CTRL-Z is the delimiter between string1 and string2.

NOTES:

As EDLIN finds each occurrence of string1, it replaces string1 with string2. EDLIN displays each line in which it performs a replacement operation. If a line contains two or more replacements of string1 with string2, then EDLIN displays the line once for each occurrence. When EDLIN replaces all occurrences of string1 in the specified range with string2, the R command is terminated and the asterisk prompt reappears.

If you include a second string as a replacement, then you must separate string1 from string2 with a CTRL-Z. You must end string2 with either a CTRL-Z RETURN combination or with a simple RETURN.

If you omit string1, then Replace takes the old string1 as its value. If there is no old string1, that is, this is the first replace, then EDLIN terminates the replacement process immediately. If you omit string2, you may end string1 with a RETURN. If you omit the line1 (as in ,line2), the first line defaults to the line after the current line. If you omit line2 (as in line1 or line1,), line2 defaults to #. Therefore, this is the same as line1,#. Remember that # indicates the line after the last line of the file.

If you end string1 with a CTRL-Z and there is no string2, EDLIN interprets string2 as an empty string and becomes the new replace string. For example:

R<string2> CTRL-Z RETURN

deletes occurrences of <string1>, but

R<string1> RETURN

replaces string1 with the old string2 and

R RETURN

replaces the old string1 with the old string2. Note that “old” here refers to a previous string specified in either a Search or a Replace command.

If you include the question mark option, the Replace command stops at each line which contains a string that matches string1, displays the line with string2 in place, and then displays the prompt O.K.?. If you press **Y** or the RETURN key, EDLIN completes the replacement and then searches for the next occurrence of string1. This process continues until the end of the range or until the end of the file. After you decide whether or not to replace the last occurrence of string1, EDLIN displays the asterisk prompt.

If you press any key besides **Y** or RETURN after the O.K.? prompt, EDLIN leaves that occurrence of string1 as it was in the line, and searches for the next occurrence of string1. If string1 occurs more than once in a line, EDLIN replaces each occurrence of string1 individually, and displays the O.K.? prompt after each replacement. In this way, you can check each replacement instance and avoid unwanted substitutions.

EXAMPLES:

Assume that the following file exists and is ready for editing:

- 1: This is a sample file
- 2: used to show dynamic line numbers.
- 3: See what happens when you use
- 4: Delete and Insert
- 5: (the D and I commands)
- 6: to edit text
- 7: in your file.
- 8: The insert command can place new lines
- 9: in the file; there's no problem
- 10: because the line numbers are dynamic;
- 11: they'll go all the way to 65533.

Replace, *cont.*

To replace all occurrences of string1 with string2 in a specified range, type:

2,12 Rand **or**

The result is:

4: Delete or Insert
 5: (the D or I commors)
 8: The insert commor can place new lines

Note that in the above replacement, EDLIN has made some unwanted substitutions. To correct these and to confirm each replacement, you can use a slightly different command.

In the next example, to replace only certain occurrences of string1 with string2, type:

2? R or **and**

The result is:

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

Now, type the List command (**L**) to see the result of all these changes:

.
 .
 4: Delete or Insert
 5: (The D or I commands)
 .
 8: The insert command can place new lines
 .
 .

NAME: Search

PURPOSE:

Searches the specified range of lines for a specified string of text.

SYNTAX:

[<line1>][,<line2>][?]S<string> RETURN

Where line1 is the first line in the block of text EDLIN is to search and line2 is the last line. The optional question mark causes EDLIN to stop for you to decide if this is where you want the Search command to terminate. string is the string of text you are searching for.

NOTES:

You must end the string with a RETURN. EDLIN displays the first line that matches string and this line becomes the current line. If you do not include the question mark option, the Search command terminates when EDLIN finds the first match. If no line contains a match for string, EDLIN displays the "Not found" message.

If you include the question mark option in the command, EDLIN displays the first line that contains a matching string; it then prompts you with the message O.K.?. If you press either the Y or RETURN key, the line becomes the current line and the search terminates. If you press any other key, the search continues until EDLIN finds another match, or until it searches all lines and displays the "Not found" message.

If you omit the line1 (as in ,<line2> S<string>), line1 defaults to the line after the current line. If you omit the line2 (as in <line1> S<string> or <line1>, S<string>), line2 defaults to # (the line after last line of file), which is the same as <line1>, # S<string>. If you omit string, Search uses the old string if one exists. (Note that "old" here refers to a string specified in a previous Search or Replace command.) If there is not an old string (for example, you have not used the Search or Replace commands in this session), EDLIN terminates the command immediately.

EXAMPLES:

Assume that the following file exists and is ready for editing:

- 1: This is a sample file
- 2: used to show dynamic line numbers.
- 3: See what happens when you use

Search, *cont.*

- 4: Delete and Insert
- 5: (the D and I commands)
- 6: to edit text
- 7: in your file.
- 8: The insert command can place new lines
- 9: in the file; there's no problem
- 10: because the line numbers are dynamic;
- 11: *they'll go all the way to 65533.

To search for the first occurrence of the string "and", type:

2,12 Sand

EDLIN displays the following line:

4: Delete and Insert

To get the "and" in line 5, modify the search command by typing:

, **12 Sand**

The search then continues from the line after the current line (line 4), because this Search command does not include a line1 option. The result is:

5: (the D and I commands)

To search through several occurrences of a string until the correct string is found, type:

1, ? Sand

The result is:

4: Delete and Insert
O.K.?

If you press any key (except **Y** or), the search continues, so type **N** here:

O.K.? **N**

Continue:

5: (the D and I commands)
O.K.?__

Now press **Y** to terminate the search:

O.K.? **Y**
*
—

To search for string "I" without the verification (O.K.?), type:

SI

EDLIN reports a match:

4: Delete and Insert

When you issue the Search command again, but this time by itself, EDLIN continues to search for the same string:

S

EDLIN reports another match:

5: (the D and I commands)

Enter **S** again:

S

EDLIN reports that the string is not found:

Not found

Note that string defaults to any string you specify in a previous Replace or Search command.

NAME: Transfer

PURPOSE:

Inserts (merges) the contents of <filename> into the file currently being edited at <line>.

SYNTAX:

[<line>]T<filename>

Where line is the line number at which EDLIN inserts text from the source file you designate for the filename parameter. If you omit the line option, then EDLIN uses the current line.

NOTES:

This command is useful if you want to put the contents of a file into another file or into the text you are typing. EDLIN inserts the transferred text at the line number specified by line and renumbers all succeeding lines.

NAME: Write

PURPOSE:

Writes a specified number of lines to disk from the lines that are being edited in memory. Lines are written to disk beginning with line number 1.

SYNTAX:

[<n>]W

Where n is the number of lines to write to disk.

NOTES:

This command is meaningful only if the file you are editing is too large to fit into memory. When you start EDLIN, EDLIN reads lines into memory until memory is 75% full.

To edit the remainder of your file, you must use the Write command to transfer some of the edited lines in memory to disk. Then you can load additional unedited lines into memory by using the Append command.

If you do not specify the number of lines, EDLIN transfers lines to disk until memory is 75% full. The Write command has no effect if you do not specify the number of lines and memory is already 75% full. EDLIN renumbers all lines, so that the first remaining line becomes line number 1.

Error Messages

When EDLIN finds an error, it displays one of the following error messages:

Cannot edit .BAK file—rename file

Cause: You attempted to edit a file with a filename extension of .BAK. You cannot edit .BAK files because EDLIN reserves this extension for backup copies.

Cure: If you need the .BAK file for editing purposes, you must either RENAME the file with a different extension, or COPY the .BAK file and give it a different filename extension.

No room in directory for file

Cause: When you attempted to create a new file, either the file directory was full or you specified an illegal disk drive or an illegal filename.

Cure: Check the command line that started EDLIN for illegal filename and illegal disk drive entries. If the command is no longer on the screen and if you have not yet typed a new command, you can recover the EDLIN start command by pressing the F3 key.

If this command line contains no illegal entries, run the MS-DOS 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.

Entry Error

Cause: The last command you typed contained a syntax error.

Cure: Retype the command with the correct syntax and press RETURN.

Line too long

Cause: During a Replace command, the string given as the replacement caused the line to expand beyond the limit of 253 characters. EDLIN aborted the Replace command.

Cure: Divide the long line into two lines, then try the Replace command twice.

Disk Full—file write not completed

Cause: You gave the End command, but the disk did not contain enough free space for the whole file. EDLIN aborted the E command and returned you to the operating system. EDLIN may have written some of the file to the disk.

Cure: EDLIN has saved only a portion (if any) of the file. You should probably delete that portion of the file and restart the editing session. The file will not be available after this error. Always be sure that the disk has sufficient free space for EDLIN to write the file to disk before you begin your editing session.

Incorrect DOS version

Cause: You attempted to run EDLIN under a version of MS-DOS that was not 2.0 or higher.

Cure: You must make sure that the version of MS-DOS that you are using is 2.0 or higher.

Invalid drive name or file

Cause: You have not specified a valid drive or filename when starting EDLIN.

Cure: Specify the correct drive or filename.

Filename must be specified

Cause: You did not specify a filename when you started EDLIN.

Cure: Specify a filename.

Invalid parameter

Cause: You specified a switch other than /B when starting EDLIN.

Cure: The /B switch is the only switch option available for starting EDLIN.

Insufficient memory

Cause: There is not enough memory to run EDLIN.

Cure: You must free some memory by writing files to disk or by deleting files before restarting EDLIN.

File not found

Cause: EDLIN did not find the filename specified during a Transfer command.

Cure: Specify a valid filename when issuing a Transfer command.

Must specify destination number

Cause: You did not specify a destination line number for a Copy or Move command.

Cure: Reissue the command with a destination line number.

Not enough room to merge the entire file

Cause: There was not enough room in memory to hold the file during a Transfer command.

Cure: You must free some memory by writing some files to disk or by deleting some files before you can transfer this file.

File creation error

Cause: EDLIN cannot create its temporary file.

Cure: Check to make sure that the directory has enough space to create the temporary file. Also, make 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 Comparison Utility

File Compare
Utility

Section 8

File Comparison Utility

Introduction

It is sometimes useful to compare files on your disk. If you have copied a file and later want to compare the copies to see which one is current, you can use the MS-DOS File Comparison Utility (FC).

The File Comparison Utility compares the contents of two files. The differences between the two files can be sent to the console or to a third file. The files you are comparing may be either source files (files containing source statements of a programming language) or binary files (files produced by the MACRO-86 assembler, the MS-LINK Linker utility, or a Microsoft high-level language compiler).

FC makes the comparisons in one of two ways: on a line-by-line basis or a byte-by-byte basis. The line-by-line comparison isolates blocks of lines that are different between the two files and prints those blocks of lines. The byte-by-byte comparison displays the bytes that are different between the two files.

Limitations on Source Comparisons

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 as much as it can load into the buffer space. If no lines match

in the portions of the files in the buffer space, FC displays only the message:

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

For binary files larger than available memory, FC compares both files completely, overlaying the portion in memory with the next portion from disk. FC displays all differences in the same manner as for files that fit completely in memory.

File Specifications

All file specifications use the following syntax:

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

Where d: is the letter designating a disk drive. If you omit the drive designation, FC defaults to the operating system's (current) default drive.

filename is a one- to eight-character name of the file.

.ext is a one- to three-character filename extension.

pathname is the path to the directory that contains filename.

Using FC

The syntax of FC is as follows:

```
FC [/#] [/B] [/W] [/C] <filename1> <filename2>
```

FC matches the first file (filename1) against the second (filename2) and reports any differences between them. Both filenames can be pathnames. For example:

```
FC B:\FOO\BAR\FILE1.TXT \BAR\FILE2.TXT
```

FC takes FILE1.TXT in the \FOO\BAR directory of disk drive B and compares it with FILE2.TXT in the \BAR directory. Because this example does not specify a drive for filename2, FC assumes that the \BAR directory is on the disk in the default drive.

FC Switches

There are four switches that you can use with the File Comparison Utility:

- /#** Stands for a number from 1 to 9. This switch specifies the number of lines required to match for the files to be considered as matching again after FC finds a difference. If you do not specify this switch, it defaults to 3. FC uses this switch only in source file comparisons.
- /B** Forces a binary comparison of both files. FC compares the two files byte-to-byte, with no attempt to re-synchronize after a mismatch. FC displays the mismatches as follows:

```

—ADDRS—      F1—      F2-
xxxxxxx      yy      zz

```

where xxxxxxxx is the relative address of the pair of bytes from the beginning of the file which has an address of 00000000. The yy and zz values are the mismatched bytes from file1 and file2, respectively. FC displays a message if one of the files contains less data than the other. For example, if file1 ends before file2, then FC displays:

```

***Data left in F2***

```

If /B is not specified, FC does a line-by-line (source file) comparison.

- /W** Causes FC to compress white space (tabs and spaces) during the comparison. Thus, FC considers multiple contiguous spaces in any line as a single space. Note that although FC compresses white space, it does not ignore them. The two exceptions are beginning and ending white

space in a line, which are ignored. For example (note that an underscore represents white space in this example):

____More__data__to__be__found____

matches with

More__data__to__be__found

and with

____More____data__to__be____found____

but does not match with

____Moredata__to__be__found

FC uses this switch in source file comparisons.

/C Causes FC to consider all letters in the files uppercase letters. For example:

Much__MORE__data__IS__NOT__FOUND

matches

much__more__data__is__not__found

If you specify both the /W and /C options, then FC compresses white space and treats lowercase letters as uppercase letters. For example:

____DATA__was__found____

matches

data__was__found

FC uses this switch only in source file comparisons.

Difference Reporting

FC reports the differences between the two files you specify by displaying the first filename, followed by the lines that differ between the files, followed by the first line to match in both files. FC then displays the name of the second file followed by the lines that are different, followed by the first line that matches. The default for the number of lines to match between the files is 3. (If you want to change this default, specify the number of lines with the `/#` switch.) For example:

```
...
...
_____<filename1>
<difference>
<1st line to match file2 in file1>

_____<filename2>
<difference>
<1st line to match file1 in file2>

_____
...
...
```

FC continues to list each difference.

If there are too many differences (involving too many lines), the program simply reports that the files are different and stops.

If FC does not find any matches after finding the first difference, FC displays:

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

and returns to the MS-DOS default drive prompt (for example, `A>`).

Redirecting FC Output to a File

FC displays the differences and matches between the two files you specify on your screen unless you redirect the output to a file. You

can accomplish this in the same way you redirect the commands for MS-DOS (refer to Section 4, "Learning About Commands").

To compare FILE1 and FILE2 and then send the FC output to DIFFER.TXT, type:

```
FC FILE1 FILE2 > DIFFER.TXT RETURN
```

FC puts the differences and matches between FILE1 and FILE2 into DIFFER.TXT on the default drive.

Example 1

Assume that these two ASCII files are on disk:

ALPHA.ASM	BETA.ASM
FILE A	FILE B
A	A
B	B
C	C
D	G
E	H
F	I
G	J
H	1
I	2
M	P
N	Q
O	R
P	S
Q	T
R	U
S	V
T	4
U	5
V	W
W	X
X	Y
Y	Z
Z	

To compare the two files and display the differences on the keyboard screen, type:

FC ALPHA.ASM BETA.ASM

FC compares ALPHA.ASM with BETA.ASM and displays the differences on the screen. All other defaults remain intact. (The defaults are: do not use tabs, spaces, or comments for matches, and do a source comparison on the two files.)

The output appears as follows on the keyboard screen (the notes do not appear):

```

_____ALPHA.ASM
D
E
F
G
_____BETA.ASM
G

```

Note: ALPHA file contains DEFG, BETA contains G.

```

_____ALPHA.ASM
M
N
O
P
_____BETA.ASM
J
1
2
P

```

Note: ALPHA file contains MNO where BETA contains J12.

```

_____ALPHA.ASM
W
_____BETA.ASM
4
5
W

```

Note: ALPHA file contains W where BETA contains 45W.

Example 2

You can print the differences on the line printer using the same two source files. In this example, four successive lines must be the same to constitute a match.

Type:

FC /4 ALPHA.ASM BETA.ASM >PRN

The following output appears on the line printer:

_____ALPHA.ASM

D
E
F
G
H
I
M
N
O
P

Note: P is the 1st of
a string of 4 matches.

_____BETA.ASM

G
H
I
J
1
2
P

_____ALPHA.ASM

W

_____BETA.ASM

4
5
W

Note: W is the 1st of a
string of 4 matches.

Example 3

This example forces a binary comparison and then displays the differences on the keyboard screen using the same two source files as in the previous examples.

Type:

FC /B ALPHA.ASM BETA.ASM

The /B switch in this example forces a binary comparison. You must type this switch and any others before the filenames in the FC command line. The following display appears on the screen:

<u>_ADDRS_</u>	<u>F1</u>	<u>F2</u>
00000009	44	47
0000000C	45	48
0000000F	46	49
00000012	47	4A
00000015	48	31
00000018	49	32
0000001B	4D	50
0000001E	4E	51
00000021	4F	52
00000024	50	53
00000027	51	54
0000002A	52	55
0000002D	53	56
00000030	54	34
00000033	55	35
00000036	56	57
00000039	57	58
0000003C	58	59
0000003F	59	5A
00000042	5A	1A

*** Data left in F1***

Error Messages

When FC detects an error, it displays one or more of the following error messages:

Incorrect DOS version

You are running FC under a version of MS-DOS that is not 2.0 or higher.

Invalid parameter:<option>

One of the switches that you have specified is invalid.

File not found:<filename>

FC could not find the filename you specified.

Read error in:<filename>

FC could not read the entire file.

Invalid number of parameters

You have specified the wrong number of options on the FC command line.

The Linker Program (MS-LINK)

Library Manager

Section 9

The Linker Program (MS-LINK)

Introduction

In this section you learn about MS-LINK. It is recommended that you read the entire section before you use MS-LINK.

Note: If you are not going to compile and link programs, you do not need to read this section.

The MS-DOS linker (called MS-LINK) is a program that:

- Combines separately produced object modules into one relocatable load module—a program you can run.
- Searches library files for definitions of unresolved external references.
- Resolves external cross-references.
- Produces a listing that shows both the resolution of external references and error messages.

Overview of MS-LINK

When you write a program, you write it in source code and then pass this source code through a compiler which produces object modules. You must then pass the object modules through the link process to produce

machine language that the computer can understand directly. This machine language is in the form required for running programs.

You may wish to link (combine) several programs and run them together. Each of your programs may refer to a symbol that is defined in another object module. This reference is called an external reference.

MS-LINK combines several object modules into one relocatable load module, or Run file (called an .EXE or Executable file). As it combines modules, MS-LINK makes sure that all external references between object modules are defined. MS-LINK can search several library files for definitions of any external references that are not defined in the object modules.

MS-LINK also produces a List file that shows external references it resolves, and it also displays any error messages.

To improve performance MS-LINK uses available memory as much as possible. After MS-LINK uses up the available memory, it creates a temporary disk file named VM.TMP.

Figure 9-1 illustrates the various parts of the MS-LINK operation.

Definitions You'll Need to Know

Some of the terms used in this section are explained below to help you understand how MS-LINK works. Generally, if you are linking object modules compiled from BASIC, Pascal, or another high-level language, you will not need to know these terms. If you are writing and compiling programs in assembly language, however, you will need to understand MS-LINK and the definitions described below. The MS-LINK section in the Macro Assembler Manual also contains useful information on how MS-LINK works.

MS-DOS divides memory into segments, classes, and groups. Figure 9-2 illustrates these concepts.

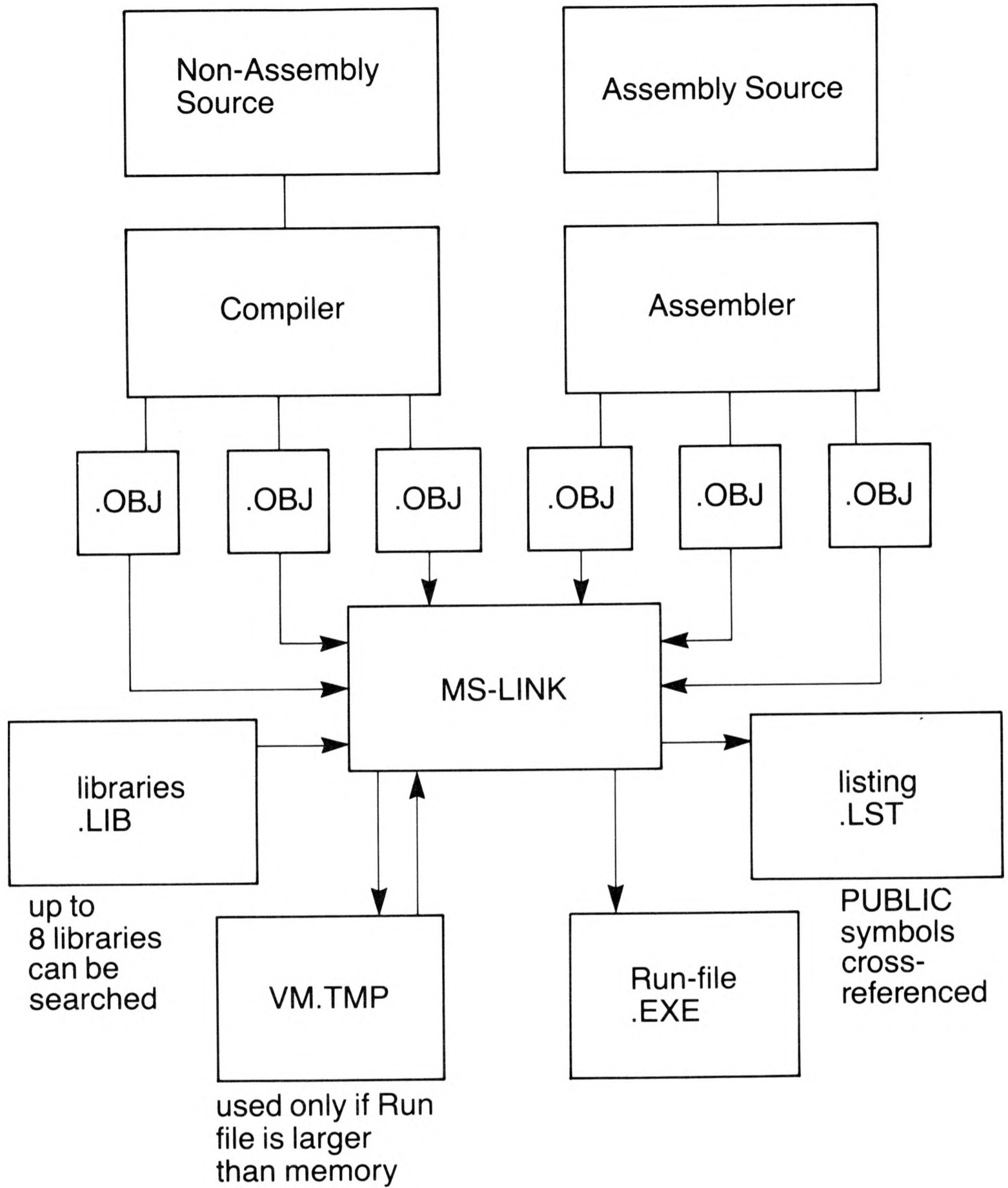
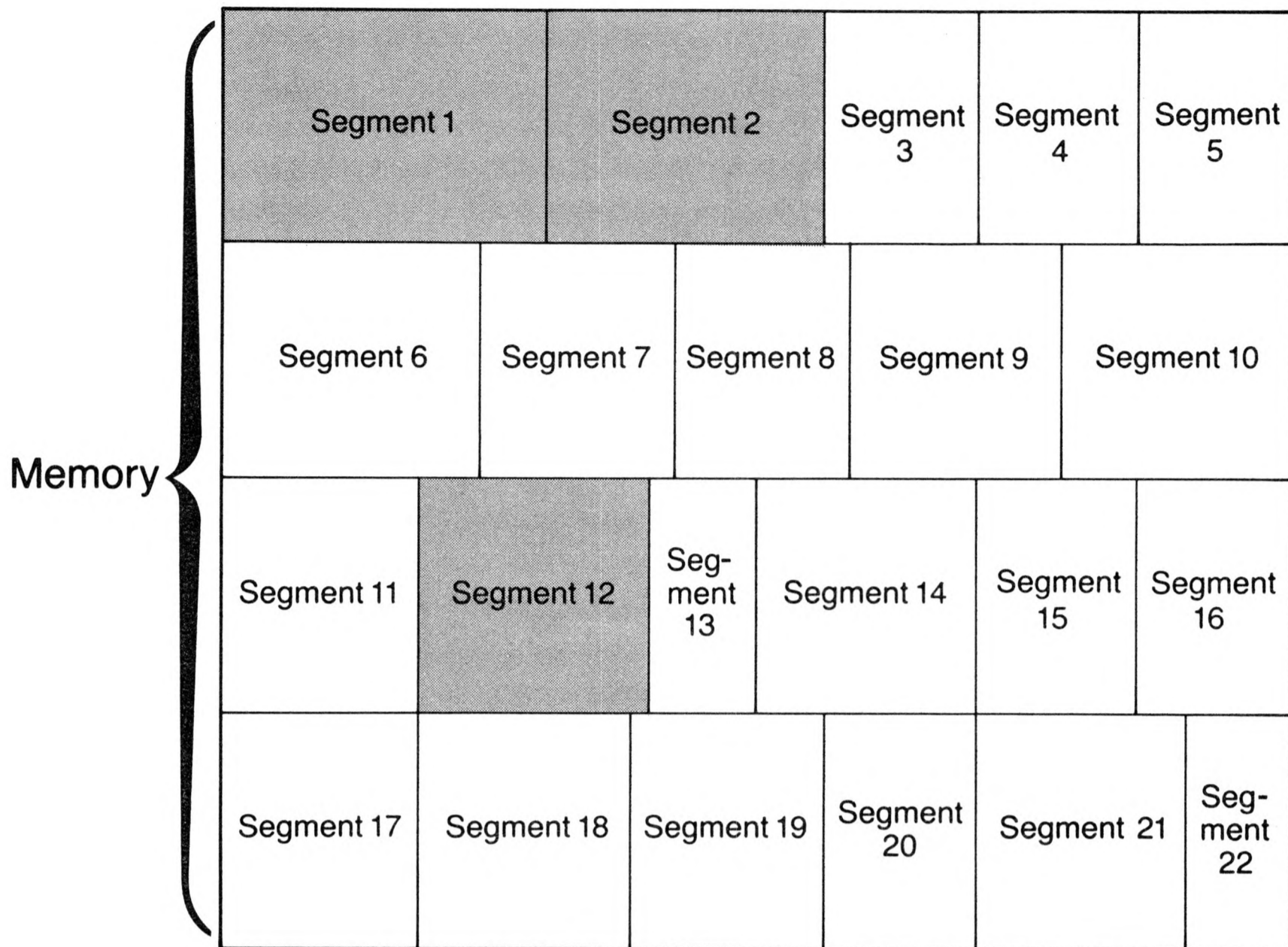


Figure 9-1: The MS-LINK operation



shaded area = a group (64K bytes addressable)

Figure 9-2: How memory is divided

Example:

	Segment Name	Segment Class Name
Segment 1	PROG.1	CODE
Segment 2	PROG.2	CODE
Segment 12	PROG.3	DATA

Note that segments 1, 2, and 12 have different segment names but may or may not have the same segment class name. Segments 1, 2, and 12 form a group with a group address of the lowest address of segment 1 (that is, the lowest address in memory).

Each segment has a segment name and a class name. MS-LINK loads all segments into memory by class name from the first segment encountered to the last. MS-DOS loads all segments assigned to the same class into memory contiguously.

During processing, MS-LINK references segments by their addresses in memory (where they are located). MS-LINK does this by finding collections of segments called groups.

A group fits in a 64K-byte area of memory. The segments within a group do not need to be contiguous. The address of any group is the lowest address of the segments in that group. At link time, MS-LINK analyzes the groups, then references the segments by the address in memory of that group. A program may consist of one or more groups.

If you are writing in assembly language, you may assign the group and class names in your program. In high-level languages (BASIC, COBOL, FORTRAN, and Pascal), the compiler assigns the names automatically.

Refer to the Macro Assembler Manual for information on how to assign group and class names and on how MS-LINK combines and arranges segments in memory.

Files that MS-LINK Uses

MS-LINK can use files to perform the following tasks:

- Work with one or more input files
- Produce two output files
- Create a temporary disk file
- Search up to eight specified library files

For each type of file, the user may give a three-part file specification. The format for MS-LINK file specifications is the same as that of other disk files:

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

where d: is the drive designation. The colon is always required as part of the drive designation.

filename is any legal filename of one to eight characters.

.ext is a one- to three-character extension to the filename. The period is always required as part of the extension.

Input File Extensions

If you do not include a filename extension for the input (object) file specifications, MS-LINK assumes the following extensions by default:

.OBJ	Object
.LIB	Library

Output File Extensions

MS-LINK appends the following default extensions to the output (Run and List) files:

.EXE	Run (may not be overridden)
.MAP	List (may be overridden)

VM.TMP (Temporary) File

MS-LINK uses available memory for the link session. If the files you are linking create an output file that exceeds available memory, MS-LINK creates a temporary file, names it VM.TMP, and puts it on the disk in the default drive. If MS-LINK creates VM.TMP, it displays the following message:

```
VM.TMP has been created.  
Do not change diskette in drive, <d:>
```

After MS-LINK displays this message, you must not remove the disk from the default drive until the link session ends. If you remove the disk before the link session is complete, the operation of MS-LINK will be unpredictable, and MS-LINK might display this error message:

```
Unexpected end of file on VM.TMP
```

MS-LINK writes the contents of VM.TMP to the file you name in response to the "Run File:" prompt. VM.TMP is a working file only and MS-LINK deletes it at the end of the linking session.

Warning: Do not use VM.TMP as a filename for any file. If you have a file named VM.TMP on the default drive and MS-LINK requires the VM.TMP file, MS-LINK deletes the VM.TMP already on disk and creates a new VM.TMP. Therefore, you lose the contents of the previous VM.TMP file.

Starting MS-LINK

MS-LINK requires two types of input: a command to start MS-LINK and responses to command prompts. In addition, seven switches control MS-LINK features. Usually, you type all the commands to MS-LINK on the keyboard. As an option, you can store answers to the command prompts and any switches in a response file. You can use command characters to assist you while entering MS-LINK commands.

You may start MS-LINK in one of three ways. The first method is to type the commands in response to individual prompts. In the second method, you type all commands on the same line you use to start MS-LINK. To start MS-LINK by the third method, you must create a response file that contains all the necessary commands and then give MS-LINK the filename and its directory path when you start MS-LINK.

Summary of Methods to Start MS-LINK

Method 1	LINK
Method 2	LINK <filenames>[/switches]
Method 3	LINK @ <d:><filespec>

Method 1: Prompts

To start MS-LINK with Method 1, type:

LINK

MS-DOS loads MS-LINK into memory. MS-LINK then displays four text prompts one at a time. You must answer the prompts to command MS-LINK to perform specific tasks.

At the end of each line, you may type one or more switches, preceded by the switch character, a slash (/).

The following list summarizes the command prompts. Refer to "Command Prompts" later in this section for more information.

Prompt	Responses
Object Modules [.OBJ]:	List .OBJ files you want to link. You must separate them with blank spaces or plus signs. If a plus sign is the last character typed, the "Object Modules [.OBJ]:" prompt reappears. There is no default; this prompt requires a response.
Run File [Object-file.EXE]:	Supply the name of the file containing executable object code. The default is the first file in the list of object modules with an extension of .EXE. (You cannot change the output extension.)
List File [Object-file.MAP]:	Supply a filename for the output listing. The default filename is the name for the run file.
Libraries []:	List the library filenames you want MS-LINK to search for external references. Separate filenames with blank spaces or plus signs. If a plus sign is the last character typed, the "Libraries []:" prompt reappears. The default is to search for default libraries in the object modules. (Extensions will be changed internally to .LIB.)

Method 2: Command Line

To start MS-LINK using Method 2, type all commands on one line. The entries following LINK are responses to the command prompts. You must separate the entry fields for the different prompts with commas. Use the following syntax:

```
LINK <object-list>, <runfile>, <listfile>, <lib-list> [/switch...]
```

The entries following LINK are responses to the command prompts which have not yet appeared. You must separate the entry fields for the different prompts with commas.

Where: object-list is a list of object modules. Use plus signs or blank spaces to separate entries.

runfile is the name of the file to receive the executable output.

listfile is the name of the file to receive the listing.

lib-list is a list of library modules to search for external references.

/switch refers to optional switches, which you may place after any of the response entries (just before any of the commas or after the lib-list, as shown).

To select the default for a field, simply type a second comma with no spaces between the two commas.

Example:

```
LINK  
FUN + TEXT + TABLE + CARE/P/M,,FUNLIST,COBLIB.LIB
```

This command causes MS-DOS to load MS-LINK followed by the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ. MS-LINK then pauses (as a result of using the /P switch) and waits for you to press any key to continue. When you press a key, MS-LINK links the object modules and produces a global symbol map (the /M switch), creates a Run file with a default name of FUN.EXE, creates a List file named FUNLIST.MAP, and searches the Library file COBLIB.LIB to resolve external references.

Method 3: Response File

To start MS-LINK with Method 3, type:

```
LINK @<d:> <filespec> 
```

Where: filespec is the name of a response file. A response file contains answers to the MS-LINK prompts (shown in Method 1) and may also contain any of the switches. When naming a response file, the use of filename extensions is optional. Method 3 permits you to enter the command that starts MS-LINK from the keyboard or within a batch file without requiring you to take any further action.

To use this option, you must create a response file containing several lines of text, each of which is the response to an MS-LINK prompt. The responses must be in the same order as the MS-LINK prompts discussed in Method 1. You may type a multi-line response to the "Object Modules:" prompt or the "Libraries:" prompt by using a plus sign to continue the same response onto the next line.

Use switches and command characters in the response file the same way as for responses typed on the keyboard.

When you begin a session with MS-LINK, it displays each prompt in order followed by the responses from the response file. If the response file does not contain answers for all the prompts (in the form of filenames, the semicolon command character, or carriage returns), MS-LINK displays the prompt which does not have a response, then waits for you to type a legal response. When you enter a legal response, MS-LINK continues the link session.

Example:

```
FUN TEXT TABLE CARE
/PAUSE/MAP
FUNLIST
COBLIB.LIB
```

This response file tells MS-LINK to load the four object modules named FUN, TEXT, TABLE, and CARE. MS-LINK pauses before producing a public symbol map to permit you to swap disks (see the discussion under /PAUSE in the "MS-LINK Switches" section before using this feature). When you press any key, MS-LINK names the output files FUN.EXE and FUNLIST.MAP. MS-LINK then searches the library file COBLIB.LIB, and uses the default settings for the switches.

Command Characters

MS-LINK provides three command characters: the plus sign, the semicolon, and the CTRL-C sequence.

Plus Sign

Use the plus sign to separate entries and to extend the current line in response to the "Object Modules:" and "Libraries:" prompts. (You may use a blank space to separate the filenames of object modules or

libraries.) To type a large number of responses (each of which may be very long), type a plus sign and the RETURN key at the end of the line to extend the line. If the plus sign and RETURN keys are the last entries following these two prompts, MS-LINK prompts you for more module names. When the "Object Modules:" or "Libraries:" prompt appears again, continue to type responses. When you are finished entering all the modules to be linked and libraries to be searched, be sure to end the response line with a module name and a RETURN and not a plus sign followed by RETURN.

Example:

```
Object Modules [.OBJ]: FUN TEXT TABLE CARE +
Object Modules [.OBJ]: FOO + FLIPFLOP + JUNQUE +
Object Modules [.OBJ]: CORSAIR
```

Semicolon

To select default responses to the remaining prompts, use a semicolon followed immediately by a carriage return at any time after the first prompt ("Run File:"). This feature saves time and overrides the need to press a series of RETURN keys.

Note: After you type the semicolon and enter it by pressing the RETURN key, you can no longer respond to any of the prompts for that link session. Therefore, do not use the semicolon to skip some prompts. To skip prompts, use the RETURN key.

Example:

```
Object Modules
[.OBJ]: FUN TEXT TABLE CARE
Run Module [FUN.EXE]: ;
```

No other prompts appear after you use the semicolon, and MS-LINK uses the default values (including FUN.MAP for the List file).

CTRL-C

Use the CTRL-C sequence to abort the link session at any time. If you type an erroneous response, such as the wrong filename or an incorrectly spelled filename, you must press CTRL-C to exit from MS-LINK and then restart MS-LINK. If you make a typing error but have not pressed the RETURN key, you may delete the erroneous characters with the backspace key, but for that line only.

Command Prompts

MS-LINK asks you for responses to four text prompts. When you type a response to a prompt and press RETURN, the next prompt appears. When you answer the last prompt, MS-LINK begins linking without further command. When the link session is finished, MS-LINK returns to the operating system. The operating system prompt appears only if MS-LINK has finished successfully. If the link session is unsuccessful, MS-LINK displays the appropriate error message.

MS-LINK prompts you for the names of Object, Run, and List files, and for Libraries. The prompts appear in the order listed in this section. Text shown in square brackets ([]) following the prompt is the default response. The "Object Modules:" prompt has no preset filename response and requires you to type a filename but not the extension which defaults to .OBJ.

Object Modules [.OBJ]:

Type a list of the object modules you want to link. MS-LINK assumes by default that the filename extension is .OBJ. You must include the extension if an object module has any filename extension other than .OBJ. Otherwise, you may omit the extension.

You must use plus signs or blank spaces to separate the filenames of the object modules.

Remember that MS-LINK loads segments into classes in the order encountered. You can use this information to set the order in which MS-LINK reads the object modules. Refer to the Macro Assembler Manual for more information on this process.

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

Typing a filename causes MS-LINK to create a file for storing the Run (executable) file that results from the link session. All Run files receive the filename extension .EXE, even if you specify an extension other than .EXE.

If you do not type a response to the "Run File:" prompt, MS-LINK uses the first filename typed in response to the "Object Modules:" prompt as the Run filename.

Example:

```
Run File [FUN.EXE]: B:PAYROLL/P
```

This response directs MS-LINK to create the Run file PAYROLL.EXE on drive B. Also, MS-LINK pauses so that you can insert a new disk to receive the Run file.

List File [Run-Filename.MAP]:

The List file contains an entry for each segment in the input (object) modules. Each entry also shows the addressing in the Run file.

The default response is the Run filename with the default filename extension .MAP.

Libraries []:

You may enter up to eight library filenames or simply a carriage return. (A carriage return tells MS-LINK to conduct a default library search.) Library files must have been created by a library utility. (Consult the MS-LIB section of the Macro Assembler Manual for information on library files.) MS-LINK assumes by default that the filename extension is .LIB for library files.

You must separate library filenames with blank spaces or plus signs.

MS-LINK searches library files in the order listed to resolve external references. When it finds the module that defines the external symbol, MS-LINK processes that module as another object module.

If MS-LINK cannot find a specified library file on the disks in the disk drives, it displays the following message:

```
Cannot find library <library-name>  
Type new drive letter:
```

Press the letter for the drive designation (for example, **B**).

MS-LINK Switches

The seven MS-LINK switches control various MS-LINK functions. You must type switches at the end of a prompt response, regardless of which method you use to start MS-LINK. You may group switches at the end of any response, or scatter them at the end of different prompt responses. If you type more than one switch at the end of one response, you still must precede each switch with a slash (/).

You may abbreviate all switches. The only restriction is that an abbreviation must be sequential from the first letter through the last letter typed; MS-LINK does not permit spaces or transpositions. For example:

Legal	Illegal
/D	/DSL
/DS	/DAL
/DSA	/DLC
/DSALLOCA	/DSALLOCT

/DSALLOCATE

Using the /DSALLOCATE switch tells MS-LINK to load all data at the high end of the processor's Data Segment. Otherwise, MS-LINK loads all data at the low end of the Data Segment. At runtime, MS-LINK sets the DS pointer to the lowest possible address to allow the entire DS segment to be used.

Use of the /DSALLOCATE switch in combination with the default load low (that is, when you do not specify the /HIGH switch) enables an application program to dynamically allocate any available memory below the area specifically allocated within DGROUP, yet to remain addressable by the same DS pointer. Pascal and FORTRAN programs require this dynamic allocation ability.

Note: MS-LINK permits your application program to dynamically allocate up to 64K bytes (or the actual amount of memory available), less the amount allocated within DGROUP.

/HIGH

Use of the /HIGH switch causes MS-LINK to place the Run file as high as possible in memory. Otherwise, MS-LINK places the Run file as low as possible.

Important: Do not use the /HIGH switch with Pascal or FORTRAN programs.

/LINENUMBERS

The /LINENUMBERS switch tells MS-LINK to include in the List file the line numbers and addresses of the source statements in the input modules. Otherwise, MS-LINK does not include line numbers in the List file.

Note: MS-LINK cannot include line numbers unless the compiler produces object modules containing line number information.

/MAP

/MAP directs MS-LINK to list all public (global) symbols defined in the input modules. If you do not specify the /MAP switch, MS-LINK lists only errors (including undefined globals).

MS-LINK lists the symbols alphabetically. For each symbol, MS-LINK lists its value and its segment:offset location in the Run file. MS-LINK lists the symbols at the end of the List file.

/PAUSE

The /PAUSE switch causes MS-LINK to pause in the link session when it encounters the switch. Normally, MS-LINK performs the linking session from beginning to end without stopping. This switch allows the user to swap the disks before MS-LINK saves the Run (.EXE) file.

When MS-LINK encounters the /PAUSE switch, it displays the following message:

```
    About to generate .EXE file
    Change disks <hit any key>
```

MS-LINK resumes processing when you press any key.

Caution: Do not remove the disk which you specified to receive the List file, or the disk containing the VM.TMP file, if MS-LINK has created one.

/STACK:<number>

Where number represents any positive numeric value (in hexadecimal radix) up to 65536 bytes. If you use a value from 1 to 511, MS-LINK uses 512. If you do not specify the /STACK switch for a link session, MS-LINK calculates the necessary stack size automatically.

All compilers and assemblers should provide information in the object modules that enable the linker to compute the required stack size.

At least one object (input) module must contain a stack allocation statement. If not, MS-LINK displays the following error message:

WARNING: NO STACK STATEMENT

/NO

/NO is short for NODEFAULTLIBRARYSEARCH. This switch tells MS-LINK not to search the default (product) libraries in the object modules. For example, if you are linking object modules in Pascal, specifying the /NO switch tells MS-LINK not to perform an automatic search for the library named PASCAL.LIB to resolve external references.

Sample MS-LINK Session

This sample shows you the type of information that MS-LINK displays during a session.

In response to the MS-DOS prompt, type:

LINK

The system displays the following messages and prompts:

Microsoft Object Linker V.2.00
© Copyright 1982 by Microsoft Inc.

Object Modules [.OBJ]: **IO SYSINIT**

Run File [IO.EXE]:

List File [NUL.MAP]: **PRN /MAP**

Libraries [.LIB]:

1. By specifying /MAP, you get both an alphabetic listing and a chronological listing of public symbols.
2. By responding PRN to the "List File:" prompt, you can redirect your output to the printer.
3. By specifying the /LINE switch, MS-LINK gives you a listing of all line numbers for all modules. (Note that the /LINE switch can generate a large volume of output.)
4. By pressing RETURN in response to the "Libraries:" prompt, you cause MS-LINK to perform an automatic library search.

Once MS-LINK 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	SYSINITSEG

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. Consult the Macro Assembler Manual for information on how to determine the actual location of relative zero, and on how to determine the absolute address of a segment.

Because you specified the /MAP switch, MS-LINK 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	SYSINIT

ADDRESS	PUBLICS BY VALUE
009F:0000	SYSINIT
009F:0005	CURRENT_DOS_LOCATION
009F:0009	FINAL_DOS_LOCATION
009F:000B	DEVICE_LIST
009F:000F	MEMORY_SIZE
009F:0011	DEFAULT_DRIVE
009F:0012	BUFFERS
009F:0013	FILES

For more information on MS-LINK, refer to the Macro Assembler Manual.

Error Messages

All errors cause the link session to abort. After you find and correct the cause of the error, you must rerun MS-LINK. MS-LINK displays the following error messages.

ATTEMPT TO ACCESS DATA OUTSIDE OF SEGMENT BOUNDS, POSSIBLY BAD OBJECT MODULE

There is probably a bad Object file.

BAD NUMERIC PARAMETER

A numeric value is specified in characters other than digits.

CANNOT OPEN TEMPORARY FILE

MS-LINK is unable to 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 the assembly language module is too complex. Simplify the DUP record in your assembly language program.

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

There is probably a bad Object file.

INVALID OBJECT MODULE

MS-LINK has encountered an incorrectly formed or incomplete object module(s) (such as a module whose assembly was stopped in the middle).

SYMBOL DEFINED MORE THAN ONCE

MS-LINK found two or more modules that define a single symbol name.

PROGRAM SIZE OR NUMBER OF SEGMENTS EXCEEDS CAPACITY OF LINKER

The total program size may not exceed 384K bytes and the number of segments may not exceed 255.

REQUESTED STACK SIZE EXCEEDS 64K

You must use the /STACK switch to specify a size greater than or equal to 64K bytes.

SEGMENT SIZE EXCEEDS 64K

64K bytes is the addressing system limit.

SYMBOL TABLE CAPACITY EXCEEDED

You may have used very many and/or very long names which caused the symbol table 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 10 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

MS-LINK does not cause this error; it is a disk error.

WARNING: NO STACK SEGMENT

You included the /STACK switch but none of the object modules specified contains a statement allocating stack space.

WARNING: SEGMENT OF ABSOLUTE OR UNKNOWN TYPE

There is a bad object module or you attempted to link modules that MS-LINK cannot handle (for example, an absolute object module).

WRITE ERROR IN TMP FILE

No more disk space remains to expand the VM.TMP file.

WRITE ERROR ON RUN FILE

There is not enough disk space for the Run file.

Section 10

Library Manager

Features of MS-LIB

Microsoft LIB is a library manager. With MS-LIB, you can:

- Create and modify library files that you use with Microsoft's MS-LINK linker utility.
- Add object files to a library.
- Delete modules from a library.
- Extract modules from a library and place the extracted modules into separate object files.

MS-LIB can create either general or special libraries, for a variety of programs or for specific programs. With MS-LIB you can create a library containing many programs, or you can create a library for one program only. The result is fast linking and more efficient execution for a language compiler or for one program.

You can modify individual modules within a library by extracting the modules, making changes, and then adding the modules to the library again. You can also replace an existing module with a different module or with a new version of an existing module.

The command scanner in MS-LIB is the same as that in MS-LINK. MS-LIB prompts you for commands that you have not supplied.

Overview of MS-LIB Operation

MS-LIB performs five library manager functions:

- Deletes modules
- Extracts a module and places it in a separate object file
- Appends an object file as a module of a library
- Replaces a module in the library file with a new module
- Creates a library file

During each library session, MS-LIB deletes or extracts modules, then appends new ones to the library file. MS-LIB reads each module into memory, checks it for consistency, and writes it back to the file. If you delete a module, MS-LIB reads that module into memory but does not write it back to the file. When MS-LIB writes a module to disk, it places that module at the end of the last module written. This procedure effectively “closes up” the disk space to conserve space and permit more library files per disk.

MS-LIB appends any new modules to the end of a library file after it reads the file into memory. Finally, MS-LIB creates the index, which MS-LINK uses to find modules and symbols in the library file. MS-LIB displays a cross-reference listing of the PUBLIC symbols in the library, if you request such a listing.

Example:

```
LIBx PASCAL-HEAP + HEAP;
```

This command first deletes the library module HEAP from the library file, then adds the file HEAP.OBJ as the last module in the library. Note that the replace function is simply the delete-append functions in succession. Also note that you can specify delete, append, or extract functions in any order. This order of execution prevents confusion in MS-LIB when a new version of a module replaces a version in the library file.

Figure 10-1 illustrates the MS-LIB operation.

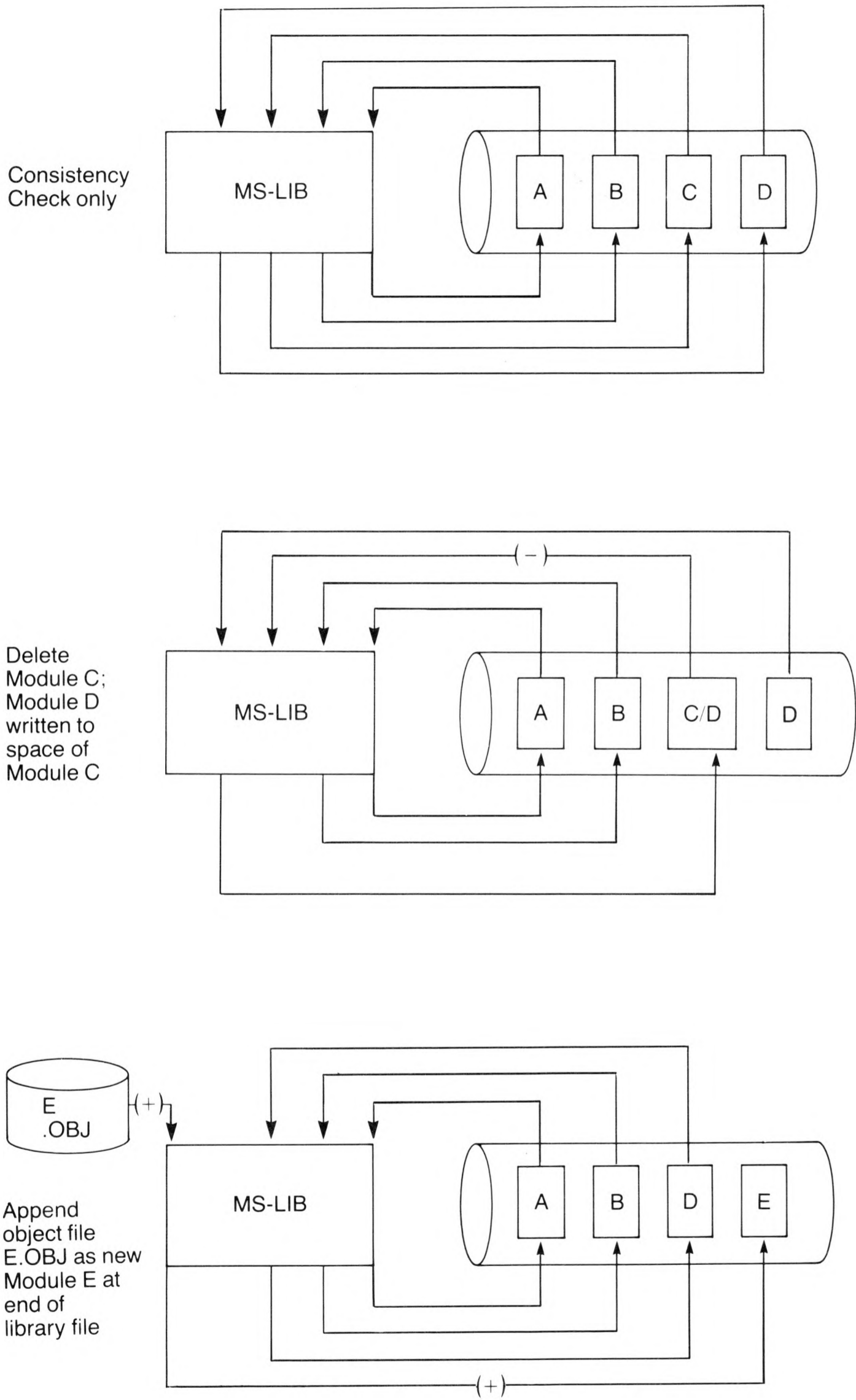
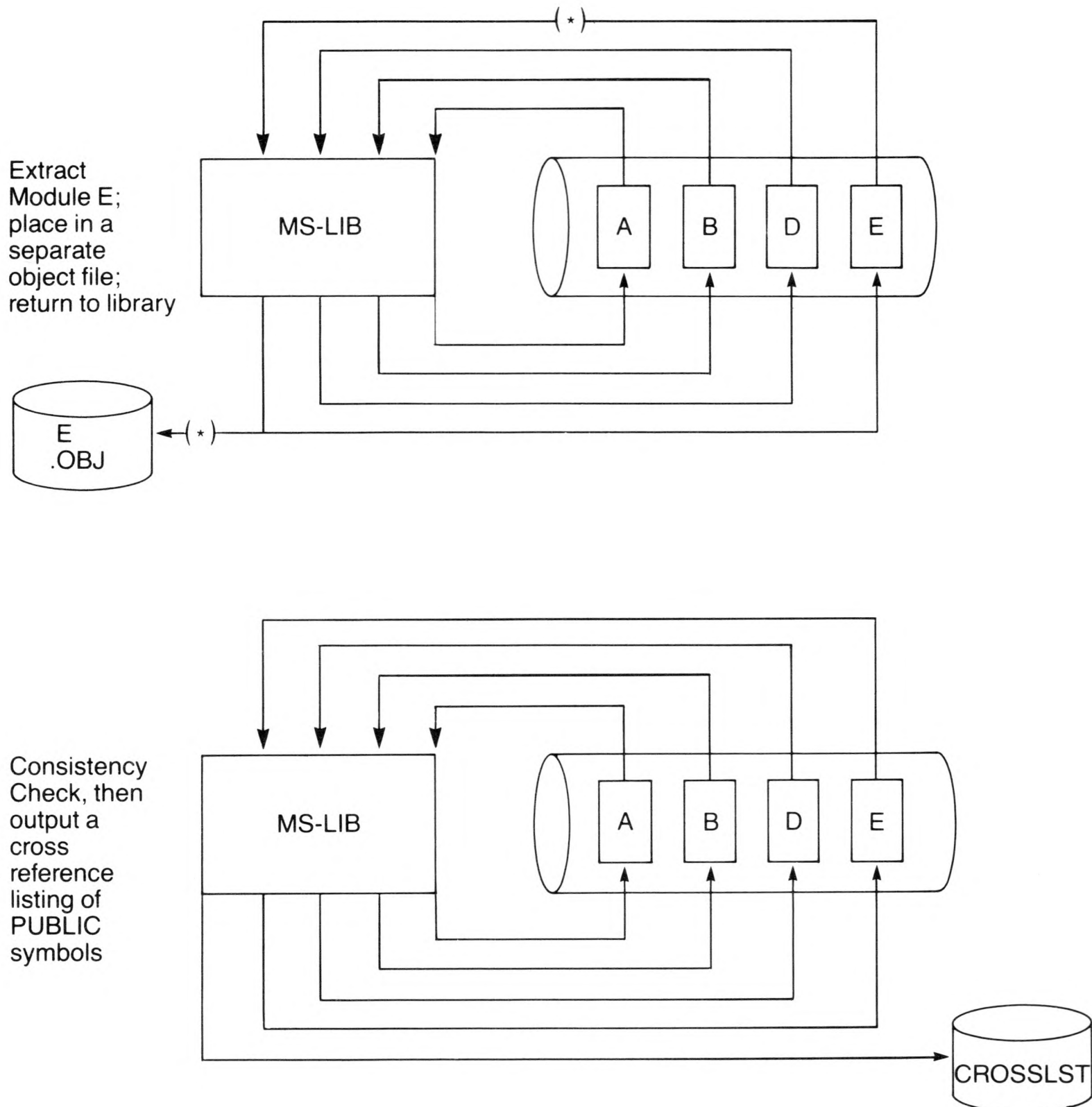


Figure 10-1: MS-LIB operation

Figure 10-1: MS-LIB operation, cont'd



Running MS-LIB

Running MS-LIB requires two types of commands: a command to start MS-LIB and responses to command prompts. Usually, you type all the commands to MS-LIB on a command line or in response to an MS-LIB prompt. As an option, you may store answers to the command prompts in a response file. You can use command characters to assist you while entering MS-LIB commands.

Starting MS-LIB

There are three ways to start MS-LIB. The first method requires you to type the commands in response to individual prompts. In the second method, you type all commands on the same line you use to start MS-LIB. As a third method, you can create a response file that contains all the necessary commands.

Summary of Methods to Start MS-LIB

Method 1: LIB

Method 2: LIB <library> <operations>, <listing>

Method 3: LIB @<d:><filespec>

Method 1: Prompts

To start MS-LIB with method 1, type:

LIB

MS-DOS loads MS-LIB into memory. MS-LIB then displays three text prompts, one at a time. You must answer the prompts to command MS-LIB to perform specific tasks.

Prompt	Response
Library File:	Enter the filename of the library you want to use. (The default is the filename extension .LIB.)
Operation:	Enter a command character(s) followed by one or more module name(s) or object filename(s). (The default is no changes. The default object filename extension is .OBJ.)
List File:	Enter a filename for a cross-reference listing file. (The default is NUL; that is, no file.)

Note: The distinction between an object file and a module (or object module) is that the file possesses a drive designation (even if it is the default drive) and a filename extension. Object modules possess neither of these parameters.

Method 2: Command Line

Type: **LIB** <library> <operations>, <listing> RETURN

The entries following LIB are responses to the command prompts. You must separate the library and operations fields and all operations entries with one of the command characters: plus, minus, or asterisk (+, -, or *). If you want a cross-reference listing, you must separate the name of the file from the last operations entry with a comma.

where: **library** is the name of a library file. MS-LIB assumes that the filename extension is .LIB, which you may override by specifying a different extension. If the filename you specify for the library parameter does not exist, MS-LIB prompts you:

Library file does not exist. Create?

Type **Y** to create a new library file. Type **N** to abort the library session.

operations is a command to delete a module, append an object file as a module, or extract a module as an object file from the library file. Use the three command characters: plus, minus, and asterisk to direct MS-LIB to append, delete, or extract modules and object files.

listing is the name of the file you want to receive the cross-reference listing of public symbols in the modules in the library. MS-LIB compiles the list after all module manipulation is complete.

If you type a library filename followed immediately by a semicolon, MS-LIB reads through the library file and performs a consistency check. MS-LIB does not change the modules in the library file.

If you type a library filename followed immediately by a comma and a listing filename, MS-LIB performs its consistency check of the library file, then produces the cross-reference listing file.

Examples:

LIB PASCAL-HEAP + HEAP;

This command line causes MS-LIB to delete the module HEAP from the library file PASCAL.LIB, then append the object file HEAP.OBJ as the last module of PASCAL.LIB (MS-LIB names the module HEAP). The MS-LIB semicolon command character indicates that MS-LIB should use the default responses for the remaining prompts.

LIB PASCAL

This example causes MS-LIB to perform only a consistency check of the library file PASCAL.LIB.

LIB PASCAL,PASCROSS.PUB

This example causes MS-LIB to perform a consistency check of the library file PASCAL.LIB, then display a cross-reference listing file named PASCROSS.PUB.

If you have many operations to perform during a library session, use the ampersand (&) command character to extend the line so that you can type additional object filenames and module names. Always include one of the command characters for operations (+, -, *) before the name of each module or object filename.

Method 3: Response File

Type: **LIB @<d:> <filespec>** RETURN

where: filespec is the name of a response file. A response file contains answers to the MS-LIB prompts. Method 3 permits you to conduct the MS-LIB session without having to respond to the individual MS-LIB prompts.

Note: Before using method 3 to start MS-LIB, you must first create a response file.

A response file has one text line for each prompt. Responses must appear in the same order as the command prompts appear.

Use command characters in the response file the same way you would use them for responses typed on the keyboard.

When the library session begins, MS-LIB displays each prompt with the responses from the response file. If the response file does not contain answers for all the prompts, MS-LIB uses the default responses. In this case, MS-LIB does not make any changes to the modules currently in the library file, and does not produce a cross-reference listing file.

If you type a library filename followed immediately by a semicolon, MS-LIB reads through the library file and performs a consistency check. In this case, MS-LIB does not make any changes to the modules in the library file.

If you type a library filename, a carriage return, a comma, and then a list filename, MS-LIB performs its consistency check of the library file, then produces the cross-reference listing file.

Example:

```
PASCAL  
+CURSOR+HEAP-HEAP*FOIBLES  
CROSSLST
```

This response file causes MS-LIB to delete the module HEAP from the PASCAL.LIB library file; extract the module FOIBLES and place it in an object file named FOIBLES.OBJ; then append the object files CURSOR.OBJ and HEAP.OBJ as the last two modules in the library. Then, MS-LIB creates a cross-reference file named CROSSLST.

Command Prompts

You command MS-LIB by typing responses to three text prompts. After you have typed your response to the current prompt, the next prompt appears. When you answer the last prompt, MS-LIB performs its library management functions without further command. When MS-LIB finishes the library session successfully, it returns you to the MS-DOS system prompt. If the library session is unsuccessful, MS-LIB displays the appropriate error message.

MS-LIB prompts you for the name of the library file, the operation(s) you want to perform, and the name you want to give to a cross-reference listing file (if any).

Library File:

Type the name of the library file that you want to manipulate. MS-LIB assumes that the filename extension is .LIB. You can override this assumption by giving a filename extension when you type the library filename. MS-LIB permits you to enter only one library filename in response to this prompt because MS-LIB can manage only one library file at a time. MS-LIB ignores additional responses, except the semicolon command character.

If you type a library filename and follow it immediately with a semicolon command character, MS-LIB performs a consistency check only, then returns to the operating system. MS-LIB displays any errors it detects in the file.

If the filename you type does not exist, MS-LIB displays the prompt:

Library file does not exist. Create?

You must type either **Y** or **N**.

Operation:

Type one of the three command characters for manipulating modules (+, -, *), followed immediately (no space) by the module name or the object filename. The plus sign appends an object file as the last module in the library file (see further discussion under the description of the plus sign in the next subsection). The minus sign deletes a module from the library file. The asterisk extracts a module from the library and places it in a separate object file, with the filename taken from the module name and a filename extension .OBJ.

When you have a large number of modules to manipulate (more than can be typed on one line), type an ampersand (&) as the last character on the line. The ampersand at the end of the line causes MS-LIB to repeat the "Operation:" prompt, which permits you to type additional module names and object filenames.

MS-LIB allows you to perform operations on modules and object files in any order you want.

List File:

If you want a public symbols cross-reference list for the modules in the library file, type the name of a file in which you want MS-LIB to place the cross-reference listing. If you do not type a filename, MS-LIB does not generate a cross-reference listing.

The response to the "List file:" prompt is a file specification. You can specify a drive (or device) designation and a filename extension with the filename. MS-LIB does not supply an extension for the list file. If you want the file to have a filename extension, you must specify it when typing the filename.

The cross-reference listing file contains two lists. The first list is an alphabetical listing of all PUBLIC symbols. Following each symbol name is the name of its module. The second list is an alphabetical list of the modules in the library. Under each module name is an alphabetical listing of the public symbols in that module.

Command Characters

MS-LIB provides six command characters. Three of the command characters are for responding to the "Operation:" prompt. The other three command characters provide you with helpful commands to simplify MS-LIB operation.

Plus Sign

Use the plus sign, followed by an object filename, to append the object file as the last module in the library named in response to the "Library File:" prompt. When MS-LIB sees the plus sign, it assumes that the filename extension is .OBJ. You may override this assumption by specifying a different filename extension.

MS-LIB strips the drive designation and the extension from the object file specification, leaving only the filename. For example, if the object file to be appended as a module to a library is

B:CURSOR.OBJ

a response to the "Operation:" prompt of

+ B:CURSOR.OBJ

causes MS-LIB to strip off the B: and the .OBJ, leaving only CURSOR. This becomes a module named CURSOR in the library.

Minus Sign

Use the minus sign, followed by a module name, to delete a module from the library file. MS-LIB then “closes up” the disk space left empty by the deletion. This cleanup action keeps the library file from growing larger than necessary. Remember that MS-LIB always adds new modules, even replacement modules, to the end of the file, not into space vacated by deleting modules.

Asterisk

Use the asterisk, followed by a module name, to extract the module from the library file and place it into a separate object file. The module will still exist in the library. (The extraction process copies the module to a separate object file.) MS-LIB makes the module name the filename by adding the default drive designation and the filename extension .OBJ. For example, if the module you want to extract is

CURSOR

and the current default disk drive is A, a response to the “Operation:” prompt of

*CURSOR

causes MS-LIB to extract the module named CURSOR from the library file and make it an object file with the file specification of:

A:CURSOR.OBJ

You can not override the drive designation and filename extension. You can, however, subsequently rename the file, giving a new filename extension; and/or copy the file to a new disk drive, giving a new filename and/or filename extension.

Semicolon

Use a single semicolon, followed immediately by a carriage return at any time after responding to the first prompt (that is, from “Library File:” on), to select default responses to the remaining prompts. This feature saves time and eliminates the need to answer additional prompts.

Note: After you type the semicolon, you can no longer respond to any of the prompts for that library session. Therefore, do not use the semicolon to skip over prompts. To skip prompts, use a carriage return. For example:

```
Library file: FUN
Operation: + CURSOR;
```

The remaining prompt does not appear, and MS-LIB uses the default value (no cross-reference file).

Ampersand

Use the ampersand to extend the current line. You can use this command character only in response to the "Operation:" prompt. The number of modules you can append is limited only by disk space. The number of modules you can replace or extract is also limited only by disk space. The number of modules you can delete is limited by the number of modules in the library file.

The line length for a response to any prompt is limited to the line length of your system. For a large number of responses to the "Operation:" prompt, place an ampersand at the end of a line. MS-LIB then displays the "Operation:" prompt again, and then you can type more responses. For example:

```
Library File: FUN
Operation: + CURSOR-HEAP + HEAP*FOIBLES&
Operation: *INIT + ASSUME + RIDE;
```

MS-LIB deletes the module HEAP; extracts the modules FOIBLES and INIT (creating two files, FOIBLES.OBJ and INIT.OBJ); then appends the object files CURSOR, HEAP, ASSUME, and RIDE. Note that MS-LIB allows you to type your "Operation:" responses in any order. You may use the ampersand character as many times as needed.

CTRL-C

Use CTRL-C to abort the library session at any time. If you type an incorrect response, such as the wrong filename or module name, or an incorrectly spelled filename or module name, you must press CTRL-C to halt MS-LIB and return to the system prompt; then you must restart MS-LIB. If you type an error and you have not pressed the RETURN key, you may delete the erroneous characters for that line only.

Here is a summary of MS-LIB command characters:

Character	Action
+	Appends an object file as the last module.
-	Deletes a module from the library.
*	Extracts a module and places it in an object file.
;	Directs MS-LIB to use default responses to remaining prompts.
&	Extends the current physical line; repeats the command prompt.
CTRL-C	Aborts the library session.

Error Messages

The following list shows MS-LIB error messages.

<symbol> is a multiply defined PUBLIC. Proceed?

Cause: Two modules define the same public symbol. This message asks you to confirm the removal of the definition of the old symbol.

Cure: Remove the PUBLIC declaration from one of the object modules and recompile or reassemble. If you respond **N**, the library is left in an indeterminate state.

Allocate error on VM.TMP

Cause: Out of disk space.

Cannot create extract file

Cause: No room in directory for extract file.

Cannot create list file

Cause: No room in directory for list file.

Cannot nest response file

Cause: @filespec in response (or indirect) file.

Cannot write library file

Cause: Out of disk space.

Close error on extract file

Cause: Out of disk space.

Error: An internal error has occurred

Contact Mindset Corporation.

Fatal Error: Cannot open input file

Cause: You mistyped an object filename.

Fatal Error: Module is not in the library

Cause: You tried to delete a module that is not in the library.

Input file read error

Cause: Bad object module or faulty disk.

Invalid object module/library

Cause: Bad object module and/or library.

Library Disk is full

Cause: No more room on disk.

Listing file write error

Cause: Out of disk space.

MS-LIB cannot open VM.TMP

Cause: There is no room for VM.TMP in disk directory.

No library file specified

Cause: No response to "Library File:" prompt.

Read error on VM.TMP

Cause: Disk not ready for read.

Symbol table capacity exceeded

Cause: Too many public symbols (about 30K characters in symbols).

Too many object modules

Cause: More than 500 object modules.

Too many public symbols

Cause: 1024 public symbols maximum.

Write error on library/extract file

Cause: Out of disk space.

Write error on VM.TMP

Cause: Out of disk space.



Instructions for
Users with
Single-Drive Systems

Disk Errors

ANSI Escape
Sequences

How to Configure
Your System

Appendix A

Instructions for Users with Single-Drive Systems

On a single-drive system, you enter the commands as you would on a two-drive system.

You should think of the single-drive system as having two drives (drive A and drive B). But instead of A and B representing two different physical drives, the A and B represent two different disks.

If you specify drive B when the “drive A disk” was last used, MS-DOS prompts you to insert the disk for drive B. For example:

```
A> COPY COMMAND.COM B: 
Insert diskette for drive B:
and strike any key when ready
      1 File(s) copied
A> _
```

If you specify drive A when the “drive B disk” was last used, MS-DOS again prompts you to change disks. This time, MS-DOS prompts you to “Insert diskette for drive A:”.

The same procedure is used for commands executed from a batch file. MS-DOS prompts you to insert the appropriate disk and press any key before it continues.

Note: The letter displayed in the system prompt represents the default drive where MS-DOS looks for any file you specify without including a drive specifier. The letter in the system prompt does not represent the last disk used.

For example, assume that A is the default drive. If the last operation performed was DIR B:, MS-DOS assumes the “drive B: disk” is still in the drive. However, the system prompt is still A>, because A is still the default drive. If you type **DIR**, MS-DOS prompts you for the “drive A: disk” because drive A is the default drive, and you did not specify another drive in the DIR command.

Appendix B

Disk Errors

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

```
<yyy> ERROR WHILE <I/O action> ON DRIVE <x>  
Abort,Ignore,Retry: __
```

In this message, yyy may be one of the following:

```
WRITE PROTECT  
BAD UNIT  
NOT READY  
BAD COMMAND  
DATA  
BAD CALL FORMAT  
SEEK  
NON-DOS DISK  
SECTOR NOT FOUND  
NO PAPER  
WRITE FAULT  
READ FAULT  
DISK
```

The I/O-action may be either of the following:

```
READING  
WRITING
```

The drive x indicates the drive in which MS-DOS detects the error.

MS-DOS waits for you to enter one of the following responses:

- A** Abort. Terminate the program requesting the disk read or write.
- I** Ignore. Ignore the bad sector and pretend the error did not occur.
- R** Retry. Repeat the operation. Use this response after you correct the source of the error (such as with NOT READY or WRITE PROTECT errors).

Usually, you will want to attempt recovery by entering responses in this order:

- R** (to try again)
- A** (to terminate the program and try a new disk)

There is one additional error message which could result from a faulty disk read or write operation:

FILE ALLOCATION TABLE BAD FOR DRIVE x

This message means that the copy in memory of one of the allocation tables has pointers to nonexistent blocks. This error can result from using an unformatted disk or from using a disk that is incorrectly formatted. If this error persists, the disk is currently unusable and you must format it prior to use.

ANSI Escape Sequences

An ANSI escape sequence is a series of characters (beginning with an escape character or keystroke) that you can use to define functions to MS-DOS. Specifically, you can reassign keys, change graphics functions, and affect cursor movement.

The ANSI escape sequences described in this appendix are not available unless you load the ANSI driver contained in the file ANSI.SYS. Use the configuration file (CONFIG.SYS) described in Appendix D to load the ANSI driver from the ANSI.SYS file.

This appendix explains the definitions of the ANSI escape sequences for MS-DOS version 2.0. At the end of this appendix are examples on how to use ANSI escape sequences.

Notes:

1. The default value is used when you do not specify an explicit value or a value of zero.
2. P_n represents a “numeric parameter”. This parameter is a decimal number specified with ASCII digits.
3. P_s represents a “selective parameter”. This parameter is any decimal number that you use to select a subfunction. You may select multiple subfunctions by separating the parameters with semicolons.

Cursor Functions

The following escape sequences affect the cursor position on the screen.

CUP - Cursor Position

ESC [Pn ; Pn H

HVP - Horizontal & Vertical Position

ESC [Pn ; Pn f

CUP and HVP move the cursor to the position specified by the parameters. The first parameter specifies the line number, and the second parameter specifies the column number. The default value is 1. When you do not specify any parameters, these escape sequences move the cursor to the home position.

CUU - Cursor Up

ESC [Pn A

This sequence moves the cursor up one line in the same column. The value of Pn determines the number of lines the cursor moves up. The default value for Pn is 1. MS-DOS ignores the CUU sequence if the cursor is already on the top line.

CUD - Cursor Down

ESC [Pn B

This sequence moves the cursor down one line in the same column. The value of Pn determines the number of lines moved. The default value for Pn is 1. MS-DOS ignores the CUD sequence if the cursor is already on the bottom line.

CUF - Cursor Forward

ESC [Pn C

The CUF sequence moves the cursor forward one column in the same line. The value of Pn determines the number of columns moved. The default value for Pn is 1. MS-DOS ignores the CUF sequence if the cursor is already in the far right column.

CUB - Cursor Backward

ESC [Pn D

This escape sequence moves the cursor back one column in the same line. The value of Pn determines the number of columns moved. The default value for Pn is 1. MS-DOS ignores the CUB sequence if the cursor is already in the far left column.

DSR - Device Status Report

ESC [6 n

The console driver displays a CPR sequence (see below) on receipt of the DSR escape sequence.

CPR - Cursor Position Report (from console driver to system)

ESC [Pn ; Pn R

The CPR sequence reports the current cursor position via standard input. The first parameter specifies the current line and the second parameter specifies the current column.

SCP - Save Cursor Postion

ESC [s

The console driver saves the current cursor position. You can restore the cursor to this position with the RCP sequence (see below).

RCP - Restore Cursor Position

ESC [u

This sequence restores the cursor to the position it had when the console driver received the SCP sequence.

Erasing

The following escape sequences affect erase functions.

ED - Erase Display

ESC [2 J

The ED sequence erases the screen and places the cursor in the home position.

EL - Erase Line

ESC [K or ESC [k

This sequence erases text from the cursor to the end of the line (including the cursor position).

Modes of Operation

The following escape sequences affect screen graphics.

SGR - Set Graphics Rendition

ESC [Ps ; ... ; Ps m

The SGR escape sequence invokes the graphic functions that you specify using the following parameter(s). The graphic functions remain in effect until the next occurrence of an SGR escape sequence.

Parameter Parameter Function

0	All Attributes off	
5	Blink on	
7	Reverse Video on	
8	Concealed on	(ISO 6429 standard)
30	Black foreground	(ISO 6429 standard)
31	Red foreground	(ISO 6429 standard)

Parameter **Parameter Function, *cont.***

32	Green foreground	(ISO 6429 standard)
33	Yellow foreground	(ISO 6429 standard)
34	Blue foreground	(ISO 6429 standard)
35	Magenta foreground	(ISO 6429 standard)
36	Cyan foreground	(ISO 6429 standard)
37	White foreground	(ISO 6429 standard)
40	Black background	(ISO 6429 standard)
41	Red background	(ISO 6429 standard)
42	Green background	(ISO 6429 standard)
43	Yellow background	(ISO 6429 standard)
44	Blue background	(ISO 6429 standard)
45	Magenta background	(ISO 6429 standard)
46	Cyan background	(ISO 6429 standard)
47	White background	(ISO 6429 standard)

SM - Set Mode

ESC [= P s h
 or ESC [= h
 or ESC [= 0 h
 or ESC [? 7 h

The SM escape sequence changes the screen width or type to one of the following parameters:

Parameter **Parameter Function**

0	40 x 25 black and white
1	40 x 25 color
2	80 x 25 black and white
3	80 x 25 color
4	320 x 200 color
5	320 x 200 black and white
6	640 x 200 black and white
7	wrap at end of line

RM - Reset Mode

ESC [= P s l
 or ESC [= l
 or ESC [= 0 l
 or ESC [? 7 l

Parameters for RM are the same as for SM (Set Mode), except that parameter 7 resets the wrap at the end of line mode.

Keyboard Reassignment

Although not part of the ANSI 3.64-1979 or ISO 6429 standard, the following keyboard reassignments are compatible with these standards.

The control sequence is:

```
ESC [ Pn ; Pn ; ... Pn p
or ESC [ "string" ; p
or ESC [ Pn ; "string" ; Pn ; Pn ; "string" ; Pn p
```

or any other combination of strings and decimal numbers.

The ANSI 3.64-1979 standard reserves the final code in the control sequence (p) for private use.

The first ASCII code in the control sequence defines which code you are mapping. The remaining numbers define the sequence of ASCII codes generated when the console driver detects this key. Note one exception: If the first code in the sequence is zero (NUL), then the first and second codes make up an extended ASCII redefinition.

Examples:

1. Reassign the Q and q key to the A and a key (and vice versa):

ESC [6 5 ; 8 1 p	A becomes Q
ESC [9 7 ; 1 1 3 p	a becomes q
ESC [8 1 ; 6 5 p	Q becomes A
ESC [1 1 3 ; 9 7 p	q becomes a

2. Reassign the F10 key to issue a DIR command followed by a carriage return:

```
ESC [ 0 ; 6 8 ; " d i r" ; 1 3 p
```

The 0;68 is the extended ASCII code for the F10 key; 13 is the decimal value of the ASCII carriage return.

How to Configure Your System

In many cases, there are installation-specific settings for MS-DOS that need to be configured at system startup. An example of this is a standard device driver, such as an online printer.

The MS-DOS configuration file (CONFIG.SYS) allows you to configure your system with a minimum of effort. With this file, you can add device drivers to your system at startup. The configuration file is simply an ASCII file that has certain commands for MS-DOS startup (boot). The boot process is as follows:

1. The system reads the disk boot sector. This sector contains enough code to read MS-DOS code.
2. The system reads the MS-DOS code.
3. MS-DOS performs a variety of initializations.
4. A system initialization routine reads the configuration file (CONFIG.SYS), if it exists, to perform device installation and other user options. Its final task is to execute the command interpreter, which finishes the MS-DOS boot process.

If there is not a CONFIG.SYS file on the MS-DOS disk, you can use the MS-DOS editor, EDLIN, to create a CONFIG.SYS file; then save it on the MS-DOS disk in your root directory.

The following list shows commands for the configuration file CONFIG.SYS:

BUFFERS = <number>

This is the number of sector buffers that will comprise the system list. It is installation-dependent. If not set, 10 is a reasonable number.

FILES = <number>

This is the number of open files that the system calls can access. It is installation-dependent. If not set, 10 is a reasonable number.

DEVICE = <filename>

This installs the device driver from filename into the system list. (See below.)

BREAK = <ON or OFF>

If you specify ON (the default is OFF), MS-DOS makes a check for CTRL-C as input for every system call. ON improves the ability to abort programs over previous versions of the MS-DOS.

SHELL = <filename>

This begins execution of the shell (top-level command processor) from filename.

A typical configuration file might look like this:

```
Buffers = 10
Files = 10
Device = \BIN\NETWORK.SYS
Break = ON
Shell = A:\BIN\COMMAND.COM A:\BIN \P
```

Note that this file sets the Buffers and Files parameters to 10. The system initialization routine searches for the filename \BIN\NETWORK.SYS to find the device that this device driver adds to the system. Make sure that you save the device file using the same pathname that you specify in the Device parameter.

This configuration file also sets the MS-DOS command EXEC to the COMMAND.COM file located on disk A in the \BIN directory. The A:\BIN tells COMMAND.COM where to look for itself when it needs to re-read from disk. The /P tells COMMAND.COM that it is the first program running on the system so that it can process the MS-DOS EXIT command.



Index

- ◀ key, 7-13
- ▶ key, 7-5
- ▼ key, 7-10

Abort, *B-2*
Addresses (MS-LINK), 9-5
ANSI escape sequences, *C-1*
 Cursor functions, *C-2*
 Erasing, *C-4*
 Keyboard reassignment, *C-6*
 Modes of operation, *C-4*
AUTOEXEC.BAT, 4-7, 5-16
Automatic Program Execution, 2-9

Backup, 3-6
Batch processing, 4-5
Binary files (FC), 8-1
BREAK, 5-6
Buffer space (FC), 8-1

CD, 3-13
Change directory, 3-13
CHDIR, 3-13, 5-7
CHKDSK, 2-11, 5-8
Class name (MS-LINK), 9-4
Classes (MS-LINK), 9-2
CLS, 5-11
Command characters, 9-10, 10-10
 Plus sign, 9-10
 Semicolon, 9-11
COMMAND.COM, 2-1
Command processor, 2-1, 4-2
Command prompts, 9-12, 10-8
 Libraries, 9-8, 9-13
 List File, 9-8, 9-13
 Object Modules, 9-8, 9-12
 Run File, 9-8, 9-12
Commands
 BREAK, 5-6
 CD, 3-13
 CHDIR, 3-13, 5-7
 CHKDSK, 2-11, 5-7
 CLS, 5-11

COPY, 3-5, 5-12
CTTY, 5-15
DATE, 5-16
DEL, 3-14, 5-17
DIR, 2-10, 3-11, 5-18
DISKCOPY, 2-7, 5-19
ECHO, 5-52
EXE2BIN, 5-21
EXIT, 5-24
External, 4-2
FIND, 4-12, 5-25
FOR, 5-53
FORMAT, 2-4, 2-10, 5-27
GOTO, 5-27
IF, 5-54
MKDIR, 3-13, 5-28
MODE, 5-29
MORE, 4-12, 5-33
PATH, 5-34
PAUSE, 4-5, 5-56
PRINT, 5-35
PROMPT, 5-37
RECOVER, 5-39
REM, 4-5, 5-40
REN, 5-41
RMDIR, 3-13, 5-42
SET, 5-43
SHIFT, 5-57
SORT, 4-12, 5-44
SYS, 5-45
TIME, 5-46
TYPE, 5-47
VER, 5-48
VERIFY, 5-49
VOL, 5-50
Compiler, 9-1
Concatenation, 5-12
Control sequence, 6-5
COPY command, 3-5, 5-12
CTRL-C, 4-4, 10-12
CTRL-S, 4-5
CTRL-Z, 4-9, 7-15
CTTY command, 5-15
Cursor functions, C-2

Data error, B-1
Date, 2-2, 5-16
Default drive, changing, 2-4
DEL, 3-14, 5-17, 7-8
Delimiters, 4-4

-
- Difference reporting (FC), 8-5
 - DIR, 2-10, 3-11, 5-18
 - Directory, 2-10, 3-6
 - Making, 3-13
 - Removing, 3-13
 - Working, 2-9
 - Directory, hierarchical, 3-6
 - DISKCOPY, 2-7, 5-19
 - Disk errors, B-1
 - Abort, B-2
 - Data, B-1
 - File Allocation Table Bad For Drive x, B-2
 - Ignore, B-2
 - Not Ready, B-1
 - Retry, B-2
 - Sector Not Found, B-1
 - Seek, B-1
 - Write Fault, B-1
 - Write Protect, B-1
 - Disks
 - Backup, 2-7
 - Drive designation, 3-2
 - Dummy parameters, 4-9

 - ECHO , 5-52
 - EDLIN commands
 - Append, 7-18
 - Copy, 7-19
 - Delete, 7-21
 - Edit, 7-23
 - End, 7-25
 - Insert, 7-26
 - List, 7-30
 - Move, 7-33
 - Page, 7-34
 - Quit, 7-35
 - Replace, 7-36
 - Search, 7-39
 - Transfer, 7-42
 - Write, 7-43
 - Erasing, C-4
 - Error messages, 10-13
 - EXE2BIN, 5-21
 - EXIT, 5-24
 - External commands, 4-2
 - External reference (MS-LINK), 9-2

 - F2 key, 7-6
 - F3 key, 7-7
 - F4 key, 7-9

F5 key, 7-12

FC, 8-1

File Allocation Table, 2-9

File Allocation Table Bad For Drive x, B-2

File specification, 3-2

Filename, 3-1, 4-3

Filename extension, 3-1, 4-2

Filename extensions, default, 9-7

 .EXE (MS-LINK), 9-7

 .MAP (MS-LINK), 9-7

 .OBJ (MS-LINK), 9-7

Filenames

 Illegal, 3-4

Files, 2-9

 Binary (FC), 8-1

 Comparing, 8-1

 Naming, 3-1

 Source (FC), 8-1

Filespec, 4-3

Filters, 4-12

FIND, 4-12, 5-25

FOR, 5-53

FORMAT command, 2-4, 2-10, 5-27

GOTO command, 5-54

Groups (MS-LINK), 9-2

Hidden files, 1-4

Hierarchical directory, 3-6

Ignore, B-2

Illegal filenames, 3-4

Input, 4-11

INS key, 7-11

Internal commands, 4-1

Keyboard reassignment, C-6

Keys

 ◀, 7-13

 ▶, 7-5

 ▼, 7-10

 DEL, 7-8

 F2, 7-6

 F3, 7-7

 F4, 7-9

 F5, 7-12

 INS, 7-11

Library file, 10-2, 10-9

LINK (FC), 8-1

List file, 10-10
Loading MS-DOS, 2-1
MKDIR command, 3-13, 5-28
MODE command, 5-29
Modes of operation, C-4
MORE command, 4-12, 5-33

Not Ready error, B-2

Object modules, 9-8, 9-12
Options, 4-3
Output, redirection, 4-11, 8-4

Parameters, 4-10
Parent directory, 3-10
PATH command, 5-34
Pathing, 3-10
Pathname, 3-9, 4-3
PAUSE command, 4-5, 5-56
Pipes, 4-13
Piping, 4-13
PRINT command, 5-35
PROMPT command, 5-37

RECOVER command, 5-39
Redirecting output (FC), 8-5
Relative address (FC), 8-3
Relocatable load module (MS-LINK), 9-1
REM, 4-5, 5-40
REN command, 5-41
Replaceable parameters, 4-9
Reserved filenames, 3-4
Response file, 10-7
Retry, B-1
RMDIR command, 3-13, 5-42

Sector Not Found error, B-1
Seek error, B-1
Segments (MS-LINK), 9-2, 9-17
Separators, 2-2
SET command, 5-43
SHIFT command, 5-57
SORT command, 4-12, 5-44
Source code, 9-1
Source files (FC), 8-1
Special characters, 3-3
Starting (MS-LINK), 9-6
Subdirectory, 3-7

Switches, 4-3, 9-14

FC, 8-3

\# (FC), 8-3

\B (FC), 8-3

\C (FC), 8-4

\W (FC), 8-3

MS-LINK

/DSALLOCATE, 9-14

/HIGH, 9-15

/LINENUMBERS, 9-15

/MAP, 9-15

/NO, 9-16

/PAUSE, 9-15

/STACK, 9-16

SYS command, 5-45

Time, 2-2, 5-46

TYPE command, 5-47

VER command, 5-48

VERIFY command, 5-49

VM.TMP (MS-LINK), 9-6

VOL command, 5-50

Wild card characters, 3-3, 4-4

The * wild card, 3-3

The ? wild card, 3-3

Working directory, 2-9, 3-10, 5-7

Displaying, 3-12

Write Fault error, B-1

Write Protect error, B-2

