AA-EA08A-XV

Rainbow™

MS™-DOS
Version 2.11
Advanced User's Guide

digital equipment corporation

First Printing, October 1984

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

MS™-DOS is a trademark of Microsoft Corporation.

CP/M is a registered trademark of Digital Research Inc. CP/M-80 and CP/M-86 are trademarks of Digital Research Inc.

The following are trademarks of Digital Equipment Corporation:

DEC	GIGI	RSX
DECmate	MASSBUS	UNIBUS
DECsystem-10	PDP	VAX
DECSYSTEM-20	P/OS	VMS
DECUS	Professional	VT
DECwriter	Rainbow	Work Processor
DIBOL	RSTS	

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

Printed in U.S.A.

CONTENTS

FIGURES

TABLES

PREFACE

INTENDED READER

This guide is intended for the advanced user of the MS-DOS operating system. The purpose of this guide is to provide detailed information about advanced MS-DOS concepts and commands.

This guide assumes that you have:

- Installed the Rainbow computer according to the instructions in the Rainbow Installation Guide.
- Read the Rainbow MS-DOS Version 2.11 Getting Started contained in this kit.
- Read the Rainbow MS-DOS Version 2.11 User's Guide contained in this kit.

GUIDE ORGANIZATION

The Rainbow MS-DOS Version 2.11 Advanced User's Guide is organized as follows:

Chapter 1	Discusses how to use directories and paths with the MS-DOS operating system
Chapter 2	Discusses batch processing
Chapter 3	Discusses input and output redirection
Chapter 4	Explains the advanced MS-DOS commands that are used by programmers
Chapter 5	Discusses the FILE COMPARISON (FC) utility
Chapter 6	Discusses the LINK utility
Appendix A	Tells you how to configure the MS-DOS operating system for the Rainbow computer

CHAPTER 1

DIRECTORIES AND PATHS

This chapter discusses the MS-DOS directory structure and how to use file names and path names within this directory structure.

1.1 DIRECTORIES

The names of your files are kept 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 are 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. It is the directory that is automatically created when you format a disk and start putting files in it. You can create additional directories and subdirectories with some of the commands discussed in Chapter 4, "Advanced MS-DOS 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 towards the root.

Unless you take special action when you create a file, new files are 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 1-1 illustrates a typical hierarchical directory structure.

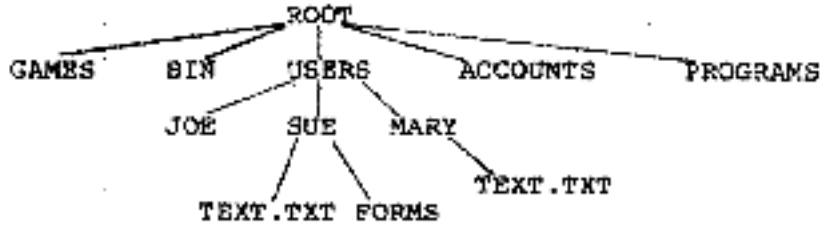


Figure 1-1: A Sample Hierarchical Directory Structure

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 Chapter 4, "Advanced MS-DOS Commands," for information on MKDIR). In this example, five subdirectories of ROOT have been created. These include:

1. A directory of games, named GAMES
2. A directory of all external commands, named BIN
3. A USER directory containing separate subdirectories for all users of the system
4. A directory containing accounting information, named ACCOUNTS
5. A directory of programs, named PROGRAMS

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

This organization of files and directories is not important if you only work with files in your own directory; but if you work with someone else or on several projects at one time, 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 \USER\SUE\FORMS
```

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

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

```
DIR \USER\MARY
```

1.2 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. This is done by giving a pathname to the file.

1.2.1 Pathnames

A simple filename is a sequence of characters that can optionally be preceded by a drive designation and followed by an extension. A pathname is a sequence of directory names followed by a simple filename, each separated from the previous one by a backslash (\).

The syntax of pathnames is:

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

If a pathname begins with a 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 \USER\SUE\TEXT.TXT.

When you are in your working directory, a filename and its corresponding pathname may be used interchangeably. Some sample names are:

\	Indicates the root directory.
\PROGRAMS	Sample directory under the root directory containing program files.

\USER\MARY\FORMS\1A	A typical full pathname. This one happens to be a file named 1A in the directory named FORMS belonging to the USER named MARY.
USER\SUE	A relative pathname; it names the file or directory SUE in subdirectory USER of the working directory. If the working directory is the root (\), (\), it names \USER\SUE.
TEXT.TXT	Name of a file or directory in the working directory.

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

. MS-DOS uses this shorthand 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 will list the files in the parent directory of your working directory.

If you type:

```
DIR ..\..
```

then MS-DOS will list the files in the parent's PARENT directory.

If you use a pathname in front of a command, (for example, E:\DOS\PRINT), the MS-DOS operating system ignores the command and returns to the operating system prompt. To use a command not in the current directory, use the PATH command.

You can use pathnames with the following MS-DOS commands:

BACKUP	FIND
CHDIR	LINK
COPY	MASM
DEBUG	MKDIR
DEL	PATH
DIR	RDCPM
EDLIN	RECOVER
ERASE	RMDIR
FC	TYPE

3.2.2 Pathing and External Commands

External commands reside on disks as program files. They must be read from the disk before they execute. (For more information on external commands, refer to the Rainbow MS-DOS Version 2.11 User's Guide).

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 tell MS-DOS in which directory these external commands reside. This is done with the PATH command.

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 terminal session. MS-DOS will now search in \BIN for the external commands. If you want to know what the current path is, type the word PATH and the current value of PATH will be printed.

For more information one the MS-DOS command PATH, refer to Chapter 4, "Advanced MS-DOS Commands."

1.2.3 Pathing and Internal Commands

Internal commands are the simplest, most commonly used commands. They execute immediately because they are incorporated into the command processor. (For more information on internal commands, refer to the Rainbow MS-DOS Version 2.11 User's Guide.)

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

The syntax of these four commands is shown below.

COPY <pathneme pathname>
If the second pathname to COPY is a directory, all files are copied into that directory.

DEL <pathname>
If the pathname is a directory, all the files in that directory are deleted. Note: The prompt "Are you sure (Y/N)?" will be displayed if you try to delete a path. Type Y to complete the command, or type N for the command to abort.

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

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

1.2.4 Displaying Your Working Directory

All commands are executed 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 \USER\JOE, when you type:

CHDIR<RETURN>

you will see:

A:\USER\JOE

This is your current drive designation plus the working directory (\USER\JOE).

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

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

.              <DIR>          8-06-84   10:09a
..             <DIR>          8-06-84   10:09a
TEXT           <DIR>          8-06-84   10:09a
FILE1   COM        5243       8-06-84    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 this output. As you can see, Joe has another directory in this tree structure named TEXT. The '.' indicates the working directory \USER\JOE, and the '..' is the shorthand notation for the parent directory \USER. FILE1.COM is a file in the \USER\JOE directory. All of these directories and files reside on the disk in drive A:.

Because files and directories are listed together (see previous display), MS-DOS does not allow you to give a subdirectory the same name as a file in that directory. For example, if you have a path \BIN\USER\JOE where JOE is a subdirectory, you cannot create a file in the USER directory named JOE.

1.2.5 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 this command has been executed by MS-DOS, 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 will automatically create the . and .. entries in the new directory.

To put files in the new directory, use the MS-DOS line editor, EDLIN. The Rainbow MS-DOS Version 2.11 User's Guide describes how to use EDLIN to create and save files.

1.2.6 How to Change 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:

```
A>CHDIR \USER
```

changes the working directory from \USER\JOE to \USER. 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.

1.2.7 How to Remove 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 it can be removed. This will prevent you from accidentally deleting files and directories. You can remove any directory by specifying its pathname. To remove the \BIN\USER\JOE directory, make sure that it has only the . and .. entries, then type:

```
RMDIR \BIN\USER\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\USER\SUE directory, type:

```
DEL \BIN\USER\SUE
```

You cannot delete the . and .. entries. They are created by MS-DOS as part of the hierarchical directory structure.

CHAPTER 2

BATCH PROCESSING

This chapter tells you how you can put a sequence of commands into a special file called a batch file. You can then execute all the commands by typing the name of the batch file. Batch processing is useful when you must type the same sequence of commands over and over again to perform a commonly used task.

2.1 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. "Batches" of your commands in such files are processed as if they were typed at a terminal. Each batch file must be named with the .BAT extension, and is executed by typing the filename without its extension.

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

BATCH PROCESSING

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. REM and PAUSE are described in detail in Chapter 4.

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 each of the lines in the .BAT file was entered at the terminal as individual commands.

Figure 2-1 illustrates the three steps used to write, save, and execute an MS-DOS batch file.

1. Write a program.

2. Assign a filename extension of .BAT and save on your directory.

3. Enter NEWDISK as a command to MS-DOS.

Figure 2-1: 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. Only the filename should be entered to execute the batch file. Do not enter the filename extension.

3. The commands in the file named <filename>.BAT are executed.

4. If you press <CTRL/C> while in batch mode, this prompt appears:

   Terminal batch job (Y/N)?

   If you press Y, the remainder of the commands in the batch file are ignored and the system prompt appears.

   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 before the next command can be read.

6. The last command in a batch file may be the name of another batch file. This allows you to call one batch file from another when the first is finished.

2.2 THE AUTOEXEC.BAT FILE

An AUTOEXEC.BAT file allows you to automatically execute programs when you start MS-DOS. Automatic program execution is useful when you want to run a specific package under MS-DOS, or 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.

When you start MS-DOS, the command processor searches the MS-DOS disk for a file named AUTOEXEC.BAT. The AUTOEXEC.BAT file is a batch file that is automatically executed each time you start the system.

If MS-DOS finds the AUTOEXEC.BAT file, the file is immediately executed by the command processor and the date and time prompts are bypassed.

If MS-DOS does not find an AUTOEXEC.BAT file when you first load the MS-DOS disk, then the date and time prompts will be issued.

Figure 2-2 illustrates how MS-DOS uses the AUTOEXEC.BAT file.

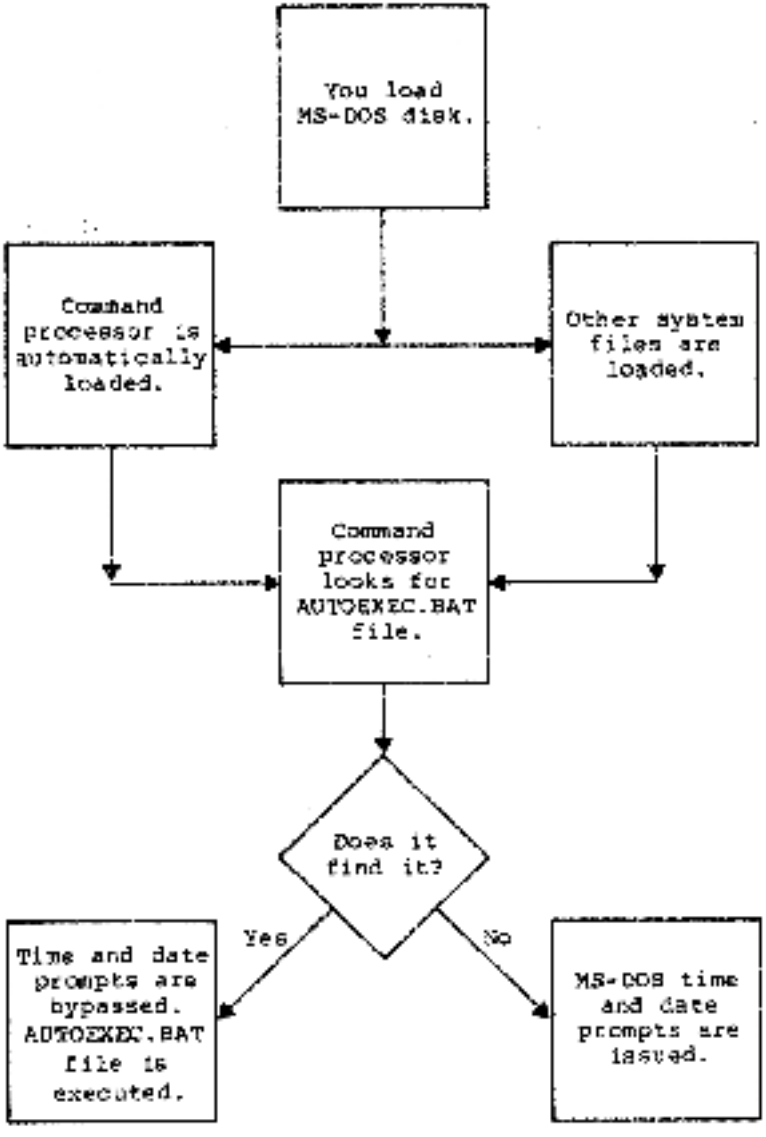


Figure 2-2: How MS-DOS Uses the AUTOEXEC.BAT File

2.2.1 How to Create an AUTOEXEC.BAT File

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

1. Type:

   ```
   COPY CON: AUTOEXEC.BAT
   ```

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

2. Now type:

   ```
   BASIC MENU
   ```

   This statement goes into the AUTOEXEC.BAT file. It tells MS-DOS to load BASIC and run the MENU program whenever MS-DOS is started.

3. Press the <CTRL/Z> key; then press the <RETURN> key to put the command BASIC MENU in the AUTOEXEC.BAT file.

4. The MENU program will now run automatically whenever you start MS-DOS.

To run your own BASIC program, enter the name of your program in place of MENU 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 will 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, since MS-DOS uses this information to keep your directory current.

2.3 CREATING A .BAT FILE WITH REPLACEMENT 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. These parameters, named %0-%9, can be replaced by values supplied when the batch file executes.

For example, when you type the command line COPY CON MYFILE.BAT, the next lines you type are copied from the console 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
```

Now, press <CTRL/Z> 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.

The dummy parameters %1 and %2 are replaced sequentially by the parameters you supply when you execute the file. The dummy parameter %0 is always replaced by the drive designator, if specified, and the filename of the batch file (for example, MYFILE).

NOTES

1. Up to 10 dummy parameters (%0-%9) can be specified. Refer to the MS-DOS command SHIFT in Chapter 4 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.

2.3.1 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, etc.

Remember that the file MYFILE.BAT consists of 3 lines:

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

To execute the MYFILE batch process, type:

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

MYFILE is substituted for %0, A:PROG1 for %1, and B:PROG2 for %2.

The result is the same as if you had 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
```

The following table illustrates how MS-DOS replaces each of the above parameters:

BATCH FILENAME	PARAMETER1 (%0) (MYFILE)	PARAMETER2 (%1) (PROG1)	PARAMETER3 (%2) (PROG2)
MYFILE	MYFILE.BAT	PROG1.MAC	PROG2.MAC PROG2.PRN

Remember that the dummy parameter %0 is always replaced by the drive designator (if specified) and the filename of the batch file.

CHAPTER 3

REDIRECTION OF INPUT AND OUTPUT

This chapter discusses input and output. It tells you how you can redirect your input and output and "filter" data. Command piping is also discussed.

3.1 INPUT AND OUTPUT

MS-DOS always assumes that input comes from the keyboard and output goes to the terminal screen. However, the flow of command input and output can be redirected. Input can come from a file rather than a terminal keyboard, and output can go to a file or to a line printer instead of to the terminal. In addition, "pipes" can be created that allow output from one command to become the input to another. Redirection and pipes are discussed in the next sections.

3.1.1 Redirecting Your Output

Most commands produce output that is sent to your terminal. 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 terminal 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), two greater-than signs (>>) can be used 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, it is created.

It is often useful to have input for a command come from a file rather than from a terminal. 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.

3.1.2 Filters

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

MS-DOS filters include FIND, MORE, and SORT. Their functions are described below:

FIND	Searches for a constant string of text in a file
MORE	Takes standard terminal output and displays it, one screen at a time
SORT	Sorts text

You can see how these filters are used in the next section.

3.1.3 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 have the output of one program sent as the input to another program. A typical case would be a program that produces output in columns. It could be desirable to have this columnar output sorted.

REDIRECTION OF INPUT AND OUTPUT

Piping is done by separating commands with the pipe separator, which is the vertical bar symbol (|). For example, the command

```
DIR | SORT
```

will give 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.

Piping can also be used when you want to 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 will create a file named DIREC.FIL on your default drive. DIREC.FIL contains a sorted listing of the directory on the default drive, since no other drive was specified in the command. To specify a drive other than the default drive, type:

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

This 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
```

will sort your directory, show it to you one screen at a time, and put --MORE-- at the bottom of your screen when there is more output to be seen.

You will find many uses for piping commands and filters.

CHAPTER 4

ADVANCED MS-DOS COMMANDS

This chapter alphabetically lists the advanced MS-DOS commands, which include commands to use with directories and paths, batch processing commands, and others.

4.1 COMMAND FORMATS

The following notation indicates how you should format the advanced MS-DOS commands:

1. You must enter any words 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 upper/lowercase; MS-DOS will convert all keywords to uppercase.

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. You must include all punctuation where shown (with the exception of square brackets), such as commas, equal signs, question marks, colons, or slashes.

4.2 MS-DOS ADVANCED COMMANDS

Table 4-1 lists the advanced MS-DOS commands:

Table 4-1: MS-DOS Advanced Commands

Command	Usage
BREAK	Sets CTRL/C check
CHDIR	Changes directories; prints working directory (CD)
CTTY	Changes console TTY
ECHO	Turns batch file echo feature on/off
EXE2BIN	Converts executable files to binary format
EXIT	Exits command and returns to lower level
FIND	Searches for a constant string of text
FOR	Provides a logical looping capability for batch files
GOTO	Provides logical flow control for batch files
IF	Provides logical flow control for batch files

Command	Usage
LDCOPY	Places the reserved system tracks onto a diskette or hard disk partition
MEDIACHK	Improves diskette performance
MKDIR	Makes a directory (MD)
MORE	Displays output one screen at a time
PATH	Sets a command search path
PAUSE	Suspends execution of a batch file
PROMPT	Designates the command prompt
RECOVER	Recovers a bad file or disk
REM	Displays a comment in a batch file
RMDIR	Removes a directory (RD)
SHIFT	Increases number of replaceable parameters in batch process
SORT	Sorts data alphabetically, forward or backward

NAME	TYPE
BREAK	Internal

PURPOSE

Sets CTRL-C check.

SYNTAX

BREAK [ON|OFF]

COMMENTS

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 <CTRL/C> 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.

If you do not specify the ON or OFF switch, MS-DOS displays the current setting of BREAK.

NAME | TYPE

CHDIR (CHANGE DIRECTORY) | Internal

SYNONYM

CD

PURPOSE

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

SYNTAX

CHDIR [pathname]

COMMENTS

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

```
CHDIR \BIN\USER\JOE\FORMS
```

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

```
CHDIR ..
```

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

CHDIR used without a pathname displays your working directory. If your working directory is \BIN\USER\JOE on drive B:, and you type CHDIR <Return>, MS-DOS will display:

```
B:\BIN\USER\JOE
```

NAME	TYPE
CTTY	Internal

PURPOSE

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

SYNTAX

CTTY <device>

COMMENTS

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).

NAME

ECHO

TYPE

Internal

PURPOSE

Turns batch echo feature on and off.

SYNTAX

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

COMMENTS

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, the current setting is displayed.

NAME	TYPE
EXE2BIN	External

PURPOSE

Converts .EXE (executable) files to binary format. This results in a saving of disk space and faster program loading.

SYNTAX

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

COMMENTS

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 no extension is specified, it defaults to .EXE. The input file is converted to .COM file format (memory image of the program) and placed in the output file. If you do not specify a drive, the drive of the input file will be used. If you do not specify an output filename, the input filename will be used. If you do not specify a filename extension in the output filename, the new file will be given 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 the initial CS:IP (Code Segment:Instruction Pointer) is specified in the .EXE file:

1. If CS:IP is not specified in the .EXE file, a pure binary conversion is assumed. If segment fixups are necessary (i.e., the program contains instructions requiring segment relocation), you will be prompted for the fixup value. This value is the absolute segment at which the program is to be loaded. 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 CS:IP is specified as 0000:100H, it is assumed that the file is to be run as a .COM file with the location pointer set at 100H by the assembler statement ORG; the first 100H bytes of the file are deleted. No segment fixups are allowed, as .COM files must be segment relocatable. Once the conversion is complete, you may rename the resulting file with a .COM extension. Then the command processor will be able to load and execute the program in the same way as the .COM programs supplied on your MS-DOS disk.

If EXE2BIN finds an error, one or more of the following error messages will be displayed:

File cannot be converted

CS:IP does not meet either of the criteria specified above, or it meets the .COM file criterion but has segment fixups. This message is also displayed if the file is not a valid executable file.

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 the finished module is to be located.

File cannot be converted

The input file is not in the correct format.

WARNING - Read error on EXE file.
Amount read less than size in header

This is a warning message only.

NAME	TYPE
EXIT	Internal

PURPOSE

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

SYNTAX

```
EXIT
```

COMMENTS

This command can be used when you are running an application program and want to start the MS-DOS command processor, then return to your program. For example, to look at a directory on drive B: while running an application program, you must start the command processor by typing COMMAND in response to the default drive prompt:

```
A>COMMAND
```

You can now type the DIR command and MS-DOS will display the directory for the default disk. When you type EXIT, you return to the previous level (your application program).

NAME	TYPE
FIND	External

PURPOSE

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

SYNTAX

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

COMMENTS

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

If no files are specified, 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 contained a match in each of the files.

/N Causes each line to be preceded by its relative line number in the file.

The string should be enclosed in quotes. 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.

When an error is detected, FIND responds with one of the following error messages:

Incorrect DOS version

FIND will only run 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 <filename>

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

FIND: Read error in <filename>

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	TYPE
FOR	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.

COMMENTS

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

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

The %%<c> variable is set sequentially to each member of <set>, and then <command> is evaluated. If a member of <set> is an expression involving * and/or ?, then the variable is set to each matching pattern from disk. In this case, only one such <item> may be in the set, and any <item> besides the first is ignored.

NOTE

The words IN, FOR, and DO must be in uppercase.

Examples:

```
FOR %%f IN ( *.ASM ) DO MASM %%f;

FOR %%f IN (FOO BAR BLECH) DO REN %%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 — TYPE

GOTO — Internal

PURPOSE

Command extension used in batch file processing.

SYNTAX

GOTO <label>

COMMENTS

GOTO causes commands to be taken from the batch file beginning with the line after the <label> definition. If no label has been defined, the current batch file will terminate.

Example:

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

will produce an infinite sequence of messages: REM looping...

Starting a line in a batch file with ':' causes the line to be ignored by batch processing. The characters following GOTO define a label, but this procedure may also be used to put in comment lines.

NAME	TYPE
IF	Internal

PURPOSE

Command extension used in batch file processing.

SYNTAX

IF <condition> <command>

COMMENTS

The parameter <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 have embedded separators.

EXIST <filename>
True if and only if <filename> 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 the <command> is executed. Otherwise, the <command> is ignored.

NOTE

The words ERRORLEVEL, EXIST, and NOT must be uppercase.

Examples:

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

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

NAME	TYPE
LDCOPY	External

PURPOSE

The LDCOPY program is used to place the MS-DOS Version 2.11 operating system's loader on a diskette or hard disk partition. The loader is a program, which is found on the first tracks of the MS-DOS Version 2.11 operating system diskette, that places the MS-DOS operating system into memory.

SYNTAX

LDCOPY <destination drv:>

COMMENTS

The destination drive is the diskette drive or hard disk drive that you want to place the loader on.

You can place the loader on a diskette or a hard disk partition that contains MS-DOS files. The LDCOPY program copies the loader and leaves the MS-DOS files intact.

The following example places the MS-DOS Version 2.11 loader on the hard disk drive E.

```
A>LDCOPY E:
```

NAME	TYPE
MEDIACHK	External

PURPOSE

Speeds access to Rainbow, MS-DOS formatted diskettes

SYNTAX

MEDIACHK [<ON|OFF>]

COMMENTS

The Rainbow computer can read a variety of MS-DOS diskette formats. The computer must first read the diskette to determine the type of MS-DOS format. If you use only Rainbow, MS-DOS formatted diskettes, use the MEDIACHK command to tell the Rainbow computer that you always use this type of diskette.

When you type the MEDIACHK command with the OFF switch, the Rainbow computer no longer checks to determine the diskette's type of MS-DOS format. This speeds the access to the diskette.

The default is MEDIACHK ON.

NAME | TYPE

MKDIR | Internal

SYNONYM

MD

PURPOSE

Makes a new directory.

SYNTAX

MKDIR <pathname>

COMMENTS

This command is used 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 \USER
```

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

```
MKDIR \USER\JOE
```

NAME | TYPE

MORE | External

PURPOSE

Sends output to console one screen at a time.

SYNTAX

MORE

COMMENTS

MORE is a filter that reads from standard input (such as a command from your terminal) 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.

Pressing the <RETURN> key will display another screen of information. This process continues until all the input data has been read.

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

```
TYPE MYFILES.COM | MORE
```

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

NAME	TYPE
PATH	Internal

PURPOSE

Sets a command path.

SYNTAX

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

COMMENTS

This command allows you to tell MS-DOS which directories should be searched for external commands after MS-DOS searches your working directory. The default value is no path.

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

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

MS-DOS will now search the \BIN\USER\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\USER\JOE;\BIN\USER\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 will print the current path. If you specify PATH ;, MS-DOS will set the NUL path, meaning that only the working directory will be searched for external commands.

NAME	TYPE
PAUSE	Internal

PURPOSE

Suspends execution of the batch file.

SYNTAX

PAUSE [comment]

COMMENTS

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 <CTRL/C>.

When the command processor encounters PAUSE, it prints:

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

If you press <CTRL/C>, another prompt will be displayed:

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

If you type Y in response to this prompt, execution of the remainder of the batch command file will be aborted and control will be returned to the operating system command level. Therefore, PAUSE can be used to break a batch file into pieces, allowing you to end the batch command file at an intermediate point.

The comment is optional and may be entered 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. An optional prompt message may be given in such cases. The comment prompt will be displayed before the "Strike a key" message.

NAME	TYPE
PROMPT	Internal

PURPOSE

Changes the MS-DOS command prompt.

SYNTAX

PROMPT [<prompt-text>]

COMMENTS

This command allows you to change the MS-DOS system prompt (for example, A>). If no text is typed, the prompt will be set 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.

When you use the PROMPT command with the $P option, be sure that you specify an existing drive. If you use this option with a non-existent drive, you could see unexpected results.

The following characters can be used in the prompt command to specify special prompts. They must all be preceded by a dollar sign ($) in the prompt command:

Specify This Character	To Get This 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
l	The '<' character
b	The '|' character
_	A CR LF sequence
s	A space (leading only)
h	A backspace
e	ASCII code X'1B' (escape)

Examples:

```
PROMPT $n
```
Sets the default drive letter prompt.

```
PROMPT Time = $t$_Date = $d
```
Sets a two-line prompt which prints:
Time = (current time)
Date = (current date)

```
PROMPT $e[7m$n:$e[m
```
Sets the prompts in inverse video mode and returns to normal video mode for other text.

NAME	TYPE
RECOVER	External

PURPOSE

Recovers a file or an entire disk containing bad sectors.

SYNTAX

RECOVER <filename | d:>

COMMENTS

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 <filename>

This will cause MS-DOS to read the file sector by sector and to skip the bad sector(s). When MS-DOS finds the bad sector(s), the sector(s) are marked and MS-DOS will no longer allocate your data to that sector.

To recover a disk, type:

RECOVER <d:>

where d: is the letter of the drive containing the disk to be recovered.

If there is not enough room on the root directory, RECOVER will print a message and store 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.

NOTE

You should use RECOVER only as a last resort. RECOVER could misinterpret data that may not be corrupted. This could lead to unpredictable results.

NAME	TYPE
REM (REMARK)	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]

COMMENTS

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

Example:

```
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:
```

NAME | TYPE

RMDIR (REMOVE DIRECTORY) | Internal

SYNONYM

RD

PURPOSE

Removes a directory from a hierarchical directory structure.

SYNTAX

RMDIR <pathname>

COMMENTS

This command removes a directory that is empty except for the . and .. shorthand symbols.

To remove the \BIN\USER\JOE directory, first issue a DIR command for that path to ensure that the directory does not contain any important files that you do not want deleted. Then type:

```
RMDIR \BIN\USER\JOE
```

The directory has been deleted from the directory structure.

NAME	TYPE
SHIFT	internal

PURPOSE

Allows access to more than 10 replaceable parameters in batch file processing.

SYNTAX

SHIFT

COMMENTS

Usually, command files are limited 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 result in the following:

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

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

NAME	TYPE
SORT	External

PURPOSE

SORT reads input from your terminal, sorts the data, then writes it to your terminal screen or files.

SYNTAX

SORT [/R] [/+n]

COMMENTS

SORT can be used, for example, to alphabetize a file by a certain column. There are two switches which allow you to select options:

/R reverse the sort; that is, sort from Z to A.

/+n sort starting with column n where n is some number. If you do not specify this switch, SORT will begin sorting from column 1.

SORT reads and sorts data in the order described in Table 4-2.

Table 4-2: SORT Table

Sort Order	ASCII Code	Character	Sort Order	ASCII Code	Character	Sort Order	ASCII Code	Character	Sort Order	ASCII Code	Character
000	000	NUL	061	061	=	096	210	Ò	193	134	SSA
001	001	SOH	062	062	>	096	242	ò	194	135	ESA
002	002	STX	063	063	?	097	211	Ó	195	136	HTS
003	003	ETX	064	064	@	097	243	ó	196	137	HTJ
004	004	EOT	065	065	A	098	212	Ô	197	138	VTS
005	005	ENQ	065	097	a	098	244	ô	198	139	PLD
006	006	ACK	066	192	À	099	213	Õ	199	140	PLU
007	007	BEL	066	224	à	099	245	õ	200	141	RI
008	008	BS	067	193	Á	100	214	Ö	201	142	SS2
009	009	HT	067	225	á	100	246	ö	202	143	SS3
010	010	LF	068	194	Â	101	080	P	203	144	DCS
011	011	VT	068	226	â	101	112	p	204	145	PU1
012	012	FF	069	195	Ã	102	081	Q	205	146	PU2
013	013	CR	069	227	ã	102	113	q	206	147	STS
014	014	SO	070	196	Ä	103	082	R	207	148	CCH
015	015	SI	070	228	ä	103	114	r	208	149	MW
016	016	DLE	071	066	B	104	083	S	209	150	SPA
017	017	DC1	071	098	b	104	115	s	210	151	EPA
018	018	DC2	072	067	C	105	223	ß	211	152	
019	019	DC3	072	099	c	106	084	T	212	153	
020	020	DC4	073	199	Ç	106	116	t	213	154	
021	021	NAK	073	231	ç	107	085	U	214	155	CSI
022	022	SYN	074	068	D	107	117	u	215	156	ST
023	023	ETB	074	100	d	108	217	Ù	216	157	OSC
024	024	CAN	075	069	E	108	249	ù	217	158	PM
025	025	EM	075	101	e	109	218	Ú	218	159	APC
026	026	SUB	076	200	È	109	250	ú	219	160	
027	027	ESC	076	232	è	110	219	Û	220	161	¡
028	028	FS	077	201	É	110	251	û	221	162	¢
029	029	GS	077	233	é	111	220	Ü	222	163	£
030	030	RS	078	202	Ê	111	252	ü	223	164	
031	031	US	078	234	ê	112	086	V	224	165	¥
032	032	SP	079	203	Ë	112	118	v	225	166	
033	033	!	079	235	ë	113	087	W	226	167	§
034	034	"	080	070	F	113	119	w	227	168	¤
035	035	#	080	102	f	114	088	X	228	169	©
036	036	$	081	071	G	114	120	x	229	170	ª
037	037	%	081	103	g	115	089	Y	230	171	«
038	038	&	082	072	H	115	121	y	231	172	
039	039	'	082	104	h	116	221	Ÿ	232	173	
040	040	(	083	073	I	116	253	ÿ	233	174	
041	041	)	083	105	i	117	090	Z	234	175	
042	042	*	084	204	Ì	117	122	z	235	176	°
043	043	+	084	236	ì	118	198	Æ	236	177	±
044	044	,	085	205	Í	118	230	æ	237	178	²
045	045	-	085	237	í	119	216	Ø	238	179	³
046	046	.	086	206	Î	119	248	ø	239	180	
047	047	/	086	238	î	120	197	Å	240	181	µ
048	048	0	087	207	Ï	120	229	å	241	182	¶
049	049	1	087	239	ï	121	091	[	242	183	·
050	050	2	088	074	J	122	092	\	243	184	
051	051	3	088	106	j	123	093	]	244	185	¹
052	052	4	089	075	K	124	094	^	245	186	º
053	053	5	089	107	k	125	095	_	246	187	»
054	054	6	090	076	L	126	096	`	247	188	¼
055	055	7	090	108	l	128	123	{	248	189	½
056	056	8	091	077	M	183	124	|	249	190	
057	057	9	091	109	m	184	125	}	250	191	¿
058	058	:	092	078	N	185	126	~	251	208	
059	059	;	092	110	n	186	127	DEL	252	222	
060	060	<	093	209	Ñ	187	128		253	240	
			093	241	ñ	188	129		254	254	
			094	215	Œ	189	130		255	255	
			094	247	œ	190	131				
			095	079	O	191	132	IND			
			095	111	o	192	133	NEL			

Examples:

This command will read the file UNSORT.TXT, reverse the sort, and then write the output to a file named SORT.TXT:

```
SORT /R <UNSORT.TXT >SORT.TXT
```

The following command will pipe the output of the directory command to the SORT filter. The SORT filter will sort the directory listing starting with column 14 (this is the column in the directory listing that contains the file size), then send the output to the console. Thus, the result of this command is a directory sorted by file size:

```
DIR | SORT /+14
```

The command

```
DIR | SORT /+14 | MORE
```

will do the same thing as the command in the previous example, except that the MORE filter will give you a chance to read the sorted directory one screen at a time.

CHAPTER 5

FILE COMPARISON UTILITY (FC)

This chapter describes how to use the File Comparison utility.

5.1 INTRODUCTION

It is sometimes useful to compare files on your disk. If you have copied a file and later want to compare 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 output to the console or to a third file. The files being compared may be either source files (files containing source statements of a programming language); or binary (files output by the MACRO-86 assembler, the MS-LINK Linker utility, or by a Microsoft high-level language compiler).

The comparisons are made in one of two ways: on a line-by-line 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.

5.1.1 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 will compare what can be loaded into the buffer space.

If no lines match in the portions of the files in the buffer space, FC will display 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. All differences are output in the same manner as those files that fit completely in memory.

5.2 FILE SPECIFICATIONS

All file specifications use the following syntax:

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

where: d: is the letter designating a disk drive. If the drive designation is omitted, 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 extension to the filename.

5.3 HOW TO USE 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. Since no drive is specified for filename2, FC assumes that the \BAR directory is on the disk in the default drive.

5.4 FC SWITCHES

There are four switches that you can use with the File Comparison Utility:

/B Forces a binary comparison of both files. The two files are compared byte-to-byte, with no attempt to re-synchronize after a mismatch. The mismatches are printed as follows:

```
--ADDRS-----F1----F2-
xxxxxxxx    yy    zz
```

(where xxxxxxxx is the relative address of the pair of bytes from the beginning of the file). Addresses start at 00000000; yy and zz are the mismatched bytes from file1 and file2, respectively. If one of the files contains less data than the other, then a message is printed out. For example, if file1 ends before file2, then FC displays:

```
***Data left in F1***
```

/# # 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 a difference has been found. If this switch is not specified, it defaults to 3. This switch is used only in source comparisons.

/W Causes FC to compress whites (tabs and spaces) during the comparison. Thus, multiple contiguous whites in any line will be considered as a single white space. Note that although FC compresses whites, it does not ignore them. The two exceptions are beginning and ending whites in a line, which are ignored.

For example (note that an underscore represents a white)

```
___More__data_to_be_found____
```

will match with

```
More_data_to_be_found
```

and with

```
_____More_____data_to_be___found____
```

but will not match with

```
___Moredata_to_be_found
```

This switch is used only in source comparisons.

/C Causes the matching process to ignore the case of letters. All letters in the files are considered uppercase letters. For example,

```
Much_MORE_data_IS_NOT_FOUND
```

will match

```
much_more_data_is_not_found
```

If both the /W and /C options are specified, then FC will compress whites and ignore case. For example,

```
:___DATA_was_found___
```

will match:

```
data_was_found
```

This switch is used only in source comparisons.

5.5 DIFFERENCE REPORTING

The File Comparison Utility 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 will continue to list each difference.

If there are too many differences (involving too many lines), the program will simply report that the files are different and stop.

If no matches are found after the first difference is found, FC will display:

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

and will return to the MS-DOS default drive prompt (for example, A>).

5.6 REDIRECTING FC OUTPUT TO A FILE

The differences and matches between the two files, you specify will be displayed on your screen unless you redirect the output to a file. This is accomplished in the same way as MS-DOS command redirection.to

To compare File1 and File2 and then send the FC output to DIFFER.TXT, type:

```
FC File1 File2 >DIFFER.TXT
```

The differences and matches between File1 and File2 will be put into DIFFER.TXT on the default drive.

5.7 EXAMPLES

Example 1:

Assume 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
N	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 terminal screeen, type:

```
FC ALPHA.ASM BETA.ASM
```

FC compares ALPHA.ASM with BETA.ASM and displays the differences on the terminal 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 will appear as follows on the terminal screen (the Notes do not appear):

```
----------ALPHA.ASM
D                       NOTE: ALPHA file
E                       contains defg.
F                       BETA contains g.
G

----------BETA.ASM
G

*****************************

----------ALPHA.ASM
M                       NOTE: ALPHA file
N                       contains mno where
O                       BETA contains j12.
P

----------BETA.ASM
J
1
2
P

*****************************

----------ALPHA.ASM
W                        NOTE: ALPHA file
                         contains w where
----------BETA.ASM       BETA contains 45w.
4
5
W
```

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 will appear on the line printer:

```
----------ALPHA.ASM
D
E
F
G
H
I
M
N           NOTE: p is the 1st of
O           a string of 4 matches.
P

----------BETA.ASM
G
H
I
J
1
2
P

--------------------------------

------------ALPHA.ASM
W                       NOTE: w is the 1st of a
------------BETA.ASM     string of 4 matches.
4
5
W
```

Example 3:

This example forces a binary comparison and then displays the differences on the terminal screen using the same two source files as were used in the previous examples.

Type:

```
FC /B ALPHA.ASM BETA.ASM
```

The /B switch in this example forces binary comparison. This switch and any others must be typed before the filenames in the FC command line. The following display should appear:

```
--ADDRS----F1---F2--
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
```

5.8 ERROR MESSAGES

When the File Comparison Utility detects an error one or more of the following error messages will be displayed:

Invalid parameter:<option>

One of the switches that you have specified is invalid.

File not found:<filename>

FC could not read 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.

Bad file

One of the files you specified is defective.

Date left in <filename>

After reaching the end of one of the files in a file comparison, the other file still has uncompared data left.

Internal error

This message indicates an internal logic error in the FC program.

CHAPTER 6

THE LINKER PROGRAM (MS-LINK)

6.1 INTRODUCTION

This chapter discusses MS-LINK. It is recommended that you read the entire chapter before you use MS-LINK.

NOTE

If you are not going to compile and link programs, you do not need to read this chapter.

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

6.2 OVERVIEW OF MS-LINK

When you write a program, you write it in source code. This source code is passed through a compiler which produces object modules. The object modules must be passed 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 resolved, and it also displays any error messages.

MS-LINK uses available memory as much as possible. When available memory is exhausted, MS-LINK creates a temporary disk file named VM.TMP.

Figure 6-1 illustrates the various parts of the MS-LINK operation.

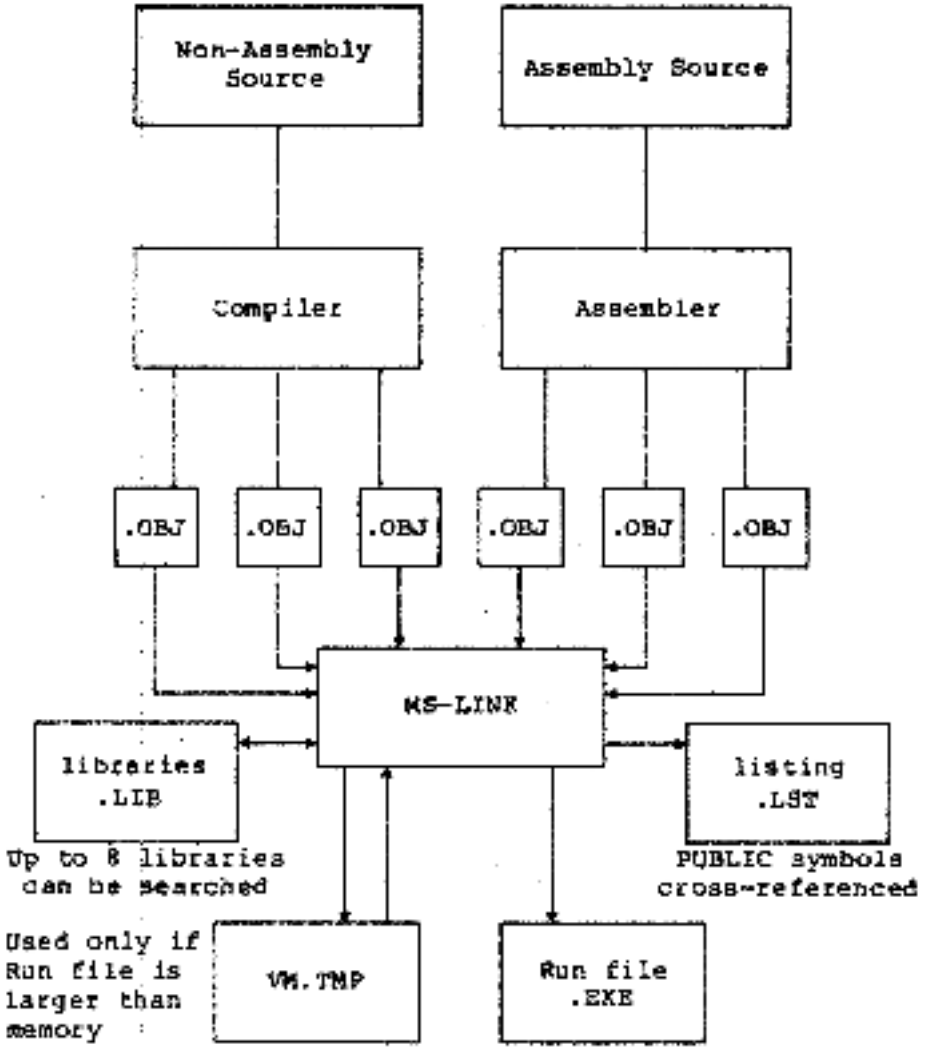
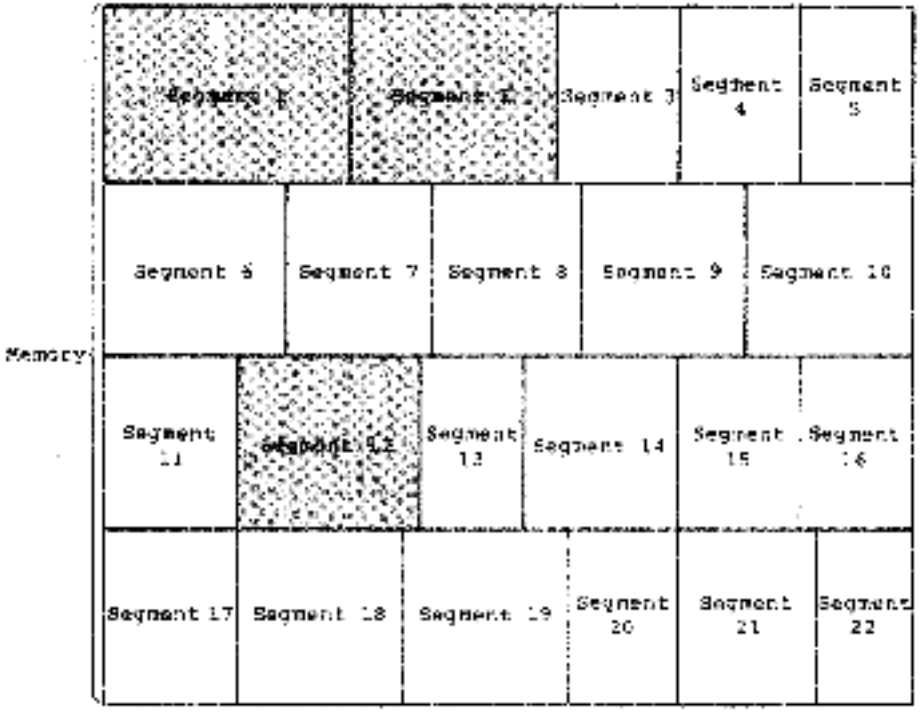


Figure 6-1: The MS-LINK Operation

6.3 DEFINITIONS YOU'LL NEED TO KNOW

Some of the terms used in this chapter are explained below to help you understand how MS-LINK works. Generally, if you are linking object modules compiled from BASIC, Pascal, or a 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.

In MS-DOS memory can be divided into segments, classes, and groups. Figure 6-2 illustrates these concepts.



shaded area = a group (64K bytes addressable)

Figure 6-2: 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 (i.e., 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. All segments assigned to the same class are loaded into memory contiguously.

During processing, MS-LINK references segments by their addresses in memory (where they are located). MS-LINK does this by finding groups of segments.

A group is a collection of segments that fit within a 64K byte area of memory. The segments do not need to be contiguous to form a group (see illustration). 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, Pascal), the naming is done automatically by the compiler.

6.4 FILES THAT MS-LINK USES

MS-LINK:

Works with one or more input files

Produces two output files

May be directed to search up to eight 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 a disk file:

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

where: d: is the drive designation. Permissible drive designations for MS-LINK are A: through O:. The colon is always required as part of the drive designation.

filename is any legal filename of one to eight characters.

.ext is one- to three-character extension to the filename. The period is always required as part of the extension.

6.4.1 Input File Extensions

If no filename extensions are given in the input (object) file specifications, MS-LINK will recognize the following extensions by default:

.OBJ	Object
.LIB	Library

6.4.2 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)
```

6.4.3 VM.TMP (Temporary) File

MS-LINK uses available memory for the link session. If the files to be linked create an output file that exceeds available memory, MS-LINK will create a temporary file, name it VM.TMP, and put it on the disk in the default drive. If MS-LINK creates VM.TMP, it will display the message:

```
VM.TMP has been created.
Do not change diskette in drive, <d:>
```

Once this message has been displayed, you must not remove the disk from the default drive until the link session ends. If the disk is removed, the operation of MS-LINK will be unpredictable, and MS-LINK might display the error message:

```
Unexpected end of file on VM.TMP
```

The contents of VM.TMP are written to the file named following the Run File: prompt. VM.TMP is a working file only and is deleted 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 will delete the VM.TMP already on disk and create a new VM.TMP. Thus, the contents of the previous VM.TMP file will be lost.

6.5 HOW TO START 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 will type all the commands to MS-LINK on the terminal keyboard. As an option, answers to the command prompts and any switches may be contained in a response file. Command characters can be used to assist you while giving commands to MS-LINK.

MS-LINK may be started in any 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 line used 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 tell MS-LINK where that file is when you start MS-LINK.

Summary of Methods to Start MS-LINK

Method 1	LINK
Method 2	LINK <filenames>[/switches]
Method 3	LINK @<filespec>

6.5.1 Method 1: Prompts

To start MS-LINK with Method 1, type:

```
LINK
```

MS-LINK will be loaded into memory. MS-LINK will then display four text prompts that appear one at a time. You 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 forward slash.

The command prompts are summarized below and described in more detail in the "Command Prompts" section.

PROMPT	RESPONSES
Object Modules [.OBJ]:	List .OBJ files to be linked. They must be separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear. There is no default; a response is required.
Run File [Object-file.EXE]:	Give filename for executable object code. The default is first-object-filename.EXE. (You cannot change the output extension.)

PROMPT	RESPONSES
List File [Run-file.MAP]:	Give filename for listing. The default is RUN filename.
Libraries [ ]:	List filenames to be searched, separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear. The default is to search for default libraries in the object modules. (Extensions will be change to .LIB.)

6.5.2 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. The entry fields for the different prompts must be separated by commas. Use the following syntax:

```
LINK (object-list>,<runfile>,<listfile>,<lib-list>[/switch...]
```

The entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas.

where: object-list is a list of object modules, separated by plus signs.

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 be searched.

/switch refers to optional switches, which may be placed following 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-LINK to be loaded, then the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ are loaded. MS-LINK then pauses (as a result of using the /P switch). MS-LINK links the object modules when you press any key, and produces a global symbol map (the /M switch); defaults to FUN.EXE Run file; creates a List file named FUNLIST.MAP; and searches the Library file COBLIB.LIB.

6.5.3 Method 3: Response File

To start MS-LINK with Method 3, type:

```
LINK @<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 the command that starts MS-LINK to be entered 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. If desired, a long response to the Object Modules: or Libraries: prompt may be typed on several lines 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 they are used for responses typed on the terminal keyboard.

When the MS-LINK session begins, each prompt will be displayed in order with 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 will display the prompt which does not have a response, then wait for you to type a legal response. When a legal response has been typed, 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 name FUN, TEXT, TABLE, and CARE. MS-LINK pauses before producing a public symbol map to permit you to swap disk (see discussion under /PAUSE in the "Switches" section before using this feature). When you press any key, the output files will be named FUN.EXE and FUNLIST.MAP. MS-LINK will search the library file COBLIB.LIB, and will use the default settings for the switches.

6.6 COMMAND CHARACTERS

MS-LINK provides three command characters.

Plus sign

Use the plus sign (+) to separate entries and to extend the current line in response to the Object Modules: and Libraries: prompts. (A blank space may be used to separate object modules.) To type a large number of responses (each may be very long), type a plus sign/<RETURN> at the end of the line to extend it. If the plus sign/<RETURN> is the last entry following these two prompts, MS-LINK will prompt you for more module names. When the Object Modules: or Libraries: prompt appears again, continue to type responses. When all the modules to be linked and libraries to be searched have been listed, be sure the response line ends with a module name and a <RETURN> and not a plus sign/<RETURN>.

Example:

```
Object Module [.OBJ]:  FUN TEXT
TABLE CARE+<RETURN>
Object Modules [.OBJ]:
FOO+FLIPFLOP+JUNQUE+<RETURN>
Object Modules [.OBJ]:
CORSAIR<RETURN>
```

Semicolon

To select default responses to the remaining prompts, use a single 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

Once the semicolon has been typed and entered (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<RETURN>
Run Module [FUN.EXE]:  ;<RETURN>
```

No other prompts will appear, and MS-LINK will use the default values (including FUN.MAP for the List file).

<CTRL/C>

Use the <CTRL/C> key 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 MS-LINK then restart MS-LINK. If the error has been typed but you have not pressed the <RETURN> key, you may delete the erroneous characters with the backspace key, but for that line only.

6.7 COMMAND PROMPTS

MS-LINK asks you for responses to four text prompts. When you have typed a response to a prompt and pressed <RETURN>, the next prompt appears. When the last prompt has been answered, MS-LINK begins linking automatically without further command. When the link session is finished, MS-LINK exits to the operating system. When the operating system prompt appears, MS-LINK has finished successfully. If the link session is unsuccessful, MS-LINK will display the appropriate error message.

MS-LINK prompts you for the names of Object, Run, and List files, and for Libraries. The prompts are listed in order of appearance. The default response is shown in square brackets ([ ]) following the prompt, for prompts which can default to preset responses. The Object Modules: prompt has no preset filename response and requires you to type a filename.

Object Modules [.OBJ]:

Type a list of the object modules to be linked. MS-LINK assumes by default that the filename extension is .OBJ. If an object module has any other filename extension, the extension must be given. Otherwise, the extension may be omitted.

Modules must be separated by plus signs (+).

Remember that MS-LINK loads segments into classes in the order encountered. You can use this information to set the order in which the object modules will be read by MS-LINK.

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

Typing a filename will 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 no response is typed 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 will pause, which allows you to 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 [ ]:

The valid responses are up to eight library filenames or simply a carriage return. (A carriage return means default library search.) Library files must have been created by a library utility.

Library filenames must be separated by 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 library file on the disks in the disk drives, it will display the message:

```
Cannot find library <library-name>
Type new drive letter:
```

Press the letter for the drive designation (for example, B).

6.8 MS-LINK SWITCHES

The seven MS-LINK switches control various MS-LINK functions. Switches must be typed at the end of a prompt response, regardless of which method is used to start MS-LINK. Switches may be grouped at the end of any response, or may be scattered at the end of several. If more than one switch is typed at the end of one response, each switch must be preceded by a forward slash (/).

All switches may be abbreviated. The only restriction is that an abbreviation must be sequential from the first letter through the last typed; no gaps or transpositions are allowed. 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 Data Segment. Otherwise, MS-LINK loads all data at the low end of the Data Segment. At runtime, the DS pointer is set 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, the /HIGH switch is not used) permits the user application to dynamically allocate any available memory below the area specifically allocated within DGroup, yet to remain addressable by the same DS pointer. This dynamic allocation is needed for Pascal and FORTRAN programs.

NOTE

Your application program may 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, line numbers are not included in the List file.

NOTE

Not all compilers produce object modules that contain line number information. In these cases, of course, MS-LINK cannot include line numbers.

/MAP

/MAP directs MS-LINK to list all public (global) symbols defined in the input modules. If /MAP is not given, MS-LINK will list only errors (including undefined globals).

The symbols are listed alphabetically. For each symbol, MS-LINK lists its value and its segment:offset location in the Run file. The symbols are listed at the end of the List file.

/PAUSE

The /PAUSE switch causes MS-LINK to pause in the link session when the switch is encountered. 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 outputs the Run (.EXE) file.

When MS-LINK encounters the /PAUSE switch, it displays the message:

```
About to generate .EXE file
Change disks <hit any key>
```

MS-LINK resumes processing when the user presses any key.

CAUTION

Do not remove the disk which will receive the List file, or the disk used for the VM.TMP file, if one has been created.

/STACK:<number>

number represents any positive numeric value (in hexadecimal radix) up to 65536 bytes. If a value from 1 to 511 is typed, MS-LINK will use 512. If the /STACK switch is not used for a link session, MS-LINK will calculate the necessary stack size automatically.

All compilers and assemblers should provide information in the object modules that allow the linker to compute the required stack size.

At least one object (input) module must contain a stack allocation statement. If not, MS-LINK will display the following error message:

WARNING: NO STACK STATEMENT

/NO

/NO is short for NODEFAULTLIBRARYSEARCH. This switch tells MS-LINK to not 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 to not automatically search the library named PASCAL.LIB to resolve external references.

6.9 SAMPLE MS-LINK SESSION

This sample shows you the type of information that is displayed during an MS-LINK session.

In response to the MS-DOS prompt, type:

```
LINK
```

The system displays the following messages and prompts (your answers are underlined):

```
   Microsoft Object Linker V2.00
(C) Copyright 1982 by Microsoft Inc.


Object Modules [.OBJ]: IO SYSINIT
Run File [IO.EXE]:
List File [NUL.MAP]: IO /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, an automatic library search is performed.

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.

Because the /MAP switch was used, 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
```

6.10 ERROR MESSAGES

All errors cause the link session to abort. After the cause has been found and corrected, MS-LINK must be rerun. The following error messages are displayed by MS-LINK:

ATTEMPT TO ACCESS DATA OUTSIDE SEGMENT BOUNDS, POSSIBLY BAD OBJECT MODULE

There is probably a bad Object file.

BAD NUMERIC PARAMETER

Numeric value is not in 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 will receive the List.MAP file.

ERROR: DUP RECORD TOO COMPLEX

DUP record in assembly language module is too complex. Simplify DUP record in 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 assembly language source and reassemble.

INPUT FILE READ ERROR

There is probably a bad Object file.

INVALID OBJECT MODULE

An object module(s) is incorrectly formed or incomplete (as when assembly is 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 size may not exceed 384k bytes and the number of segments may not exceed 255.

REQUESTED STACK SIZE EXCEEDS 64K

Specify a size greater than or equal to 64K bytes with the /STACK switch.

SEGMENT SIZE EXCEEDS 64K

64K bytes is the addressing system limit.

SYMBOL TABLE CAPACITY EXCEEDED

Very many and/or very long names were typed, exceeding 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 8 libraries.

TOO MANY PUBLIC SYMBOLS

The limit is 1024 public symbols.

TOO MANY SEGMENTS OR CLASSES

The limit is 256 (segments and classes taken together).

UNRESOLVED EXTERNALS: <list>

The external symbols listed have no defining module among the modules or library files specified.

VM READ ERROR

This is a disk error; it is not caused by MS-LINK

WARNING: NO STACK SEGMENT

None of the object modules specified contains a statement allocating stack space, but you typed the /STACK switch.

WARNING: SEGMENT OF ABSOLUTE OR UNKNOWN TYPE

There is a bad object module or an attempt has been made to link modules that MS-LINK cannot handle (e.g., 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

Usually, there is not enough disk space for the Run file.

APPENDIX A

HOW TO CONFIGURE YOUR SYSTEM

This appendix tells you how to specify MS-DOS settings that need to be configured at start-up time. An example of this is a standard device driver, such as an online printer. 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 disk boot sector is read. This contains enough code to read MS-DOS code and the BIOS.

2. The MS-DOS code and BIOS are read.

3. A variety of BIOS initializations are done.

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.

A.1 CHANGING THE CONFIG.SYS FILE

If there is not a CONFIG.SYS file on the MS-DOS disk, you can use the MS-DOS editor, EDLIN, to create a file; then save it on the MS-DOS disk in your root directory.

The following is a list of commands for the configuration file CONFIG.SYS:

BUFFERS = <number>

This is the number of sector buffers that will comprise the system list. If not set, 10 is a reasonable number.

FILES = <number>

This is the number of open files that the system calls 2FH through 57H can access. If not set, 10 is a reasonable number.

DEVICE = <filename>

This installs the device driver in <filename> into the system list. (See below.)

BREAK = <ON or OFF>

If ON is specified (the default is OFF), a check for CTRL/C as input will be made every time the system is called. 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>.

COUNTRY = <number>

This number is set to allow for international date, time, currency, and case conversion. Acceptable values are:

United States = 1 (default value)

France = 33

Spain = 34

Portugal = 35

Italy = 39

United Kingdom = 44

Germany = 49

Japan = 81

Israel = 97

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 here that the Buffers and Files parameters are set to 10. The system initialization routine will search for the filename \BIN\NETWORK.SYS to find the device that is being added to the system. This file is usually supplied on disk with your device. Make sure that you save the device file in the pathname that you specify in the Device parameter.

Release Note for MS-DOS OPERATING SYSTEM
Version 2.11

IMPORTANT - Please read these notes if you intend to use the International Features of this version of the operating system.

Appendix D of the manual entitled "MS-DOS Version 2.11 User's Guide" describes the International Features of this product.

Please note that the list of country codes on Page D-4 does not apply and should be replaced with the following list:

COUNTRY	COUNTRY CODE
USA	1
Holland	31
France	33
Finland	38
UK	44
Denmark	45
Sweden	46
Norway	47
Germany	49
Japan	81

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

Rainbow™
MS™–DOS
Version 2.11
Advanced User's Guide
AA–EA08A–XV

READER'S COMMENTS

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ First-time computer user
- ☐ Experienced computer user
- ☐ Application package user
- ☐ Programmer
- ☐ Other (please specify)

Name

Date

Organization

Street

City

State

Zip Code
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary If Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MRO1-2-L12
MARLBOROUGH, MA 01752

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line