IBM

Personal Computer Computer Language Series

Disk Operating System

by Microsoft Corp.

1502343

Personal Computer
Computer Language
Series

Disk Operating System

by Microsoft Corp.

First Edition (September 1983)
Version 2.10

Preface

Read This First

This manual explains how to use the IBM Personal Computer Disk Operating System Version 2.10 (DOS). Information is also given on how to prepare DOS and your fixed disk for use, and how to use the features of DOS Version 2.10.

This book together with the IBM *DOS Technical Reference* can be used by experienced programmers when using developed application programs. Information is provided on how to issue commands to DOS to create, edit, link, debug, and execute programs.

First Time Users

Before using your DOS diskette for the first time read the sections "About Diskettes" and "Backing Up Your DOS Diskettes" in the IBM *Disk Operating System User's Guide.*

Experienced DOS Users

Experienced application programmers will find the IBM *DOS Technical Reference* helpful.

About Your DOS Diskettes

DOS Version 2.10 is shipped on two diskettes. The first labeled "DOS" contains the DOS programs and commands. It is referred to as the DOS diskette in this manual. The second diskette labeled "DOS Supplemental Programs" contains the LINK Utility, EXE2BIN, DEBUG, and several BASIC sample programs (see the IBM Personal Computer BASIC book for information about the sample programs).

Terms Used

The terms "disk," "diskette," and "fixed disk" are used throughout this book. Where "diskette" is used, it applies only to diskette drives and diskettes. Where "fixed disk" is used, it applies only to the IBM nonremovable fixed disk drive. Where "disk" is used, it applies to both fixed disks and diskettes.

Considerations for Using Applications

If you have any of the following applications, please note there are different storage requirements when using these applications with DOS Version 2.10. Refer to Appendix D.

- Accounting Packages by BPI Systems, Inc.
- Accounting Packages Version 1.00 by Peachtree Software, Inc.
- Accounting Packages Version 1.10 by Peachtree Software, Inc.
- Arithmetic Games 1 and 2
- Asynchronous Communications Support Version 1.00
- Asynchronous Communications Support Version 2.00
- EasyWriter Version 1.10
- Fact Track
- Multiplan 1.00
- pfs:FILE
- pfs:REPORT

- SNA 3270 Emulation and RJE Support Version 1.00
- The Dow Jones Reporter Version 1.00
- Typing Tutor
- VisiCalc Version
- 3101 Emulator Version 1.00

Organization of This Manual

This manual has 8 chapters and 6 appendixes.

Chapter 1 is an introduction to the commands you can issue to DOS. These commands enable you to manage files, maintain disks, create programs, and execute programs.

Chapter 2 gives a detailed description of each command, which is listed in alphabetical order. Each description contains the purpose, format, and type of each command. Examples are given where appropriate.

Chapter 3 gives detailed instructions on how to use the fixed disk.

Chapter 4 tells you how to configure your system.

Chapter 5 describes how to use tree-structured directories.

Chapter 6 tells you how to use the Line Editor (EDLIN) program to create, alter, and display source language files and text files.

Chapter 7 describes the use of the linker (LINK) program to link programs together before execution.

Chapter 8 describes the uses of the DEBUG program in providing a controlled test environment so you can monitor and control the execution of a program to be debugged. This is done by altering, displaying, and executing object files.

Appendix A lists messages generated by the programs described in this manual.

Appendix B describes how to run compilers and the macro assembler with fixed disk.

Appendix C explains how to run the Pascal compiler with fixed disk.

Appendix D contains important information about the application programs.

Appendix E gives a brief description of all the enhancements from DOS 1.10 to DOS 2.00 and 2.10.

Appendix F discusses the editing keys you use with DOS.

Contents

Chapter 1. Using DOS

Contents

Introduction

This chapter explains how to use the DOS commands. Chapter 2 lists the commands in alphabetical order and gives a detailed description of each command.

You can use DOS commands to:

- Compare, copy, display, erase, rename files, and format fixed disks and diskettes.
- Execute system programs, such as EDLIN and DEBUG, plus your own programs.
- Set various printer and screen options.
- Request DOS to pause.
- Transfer DOS to another diskette.
- Cause printer output to be directed to the Asynchronous Communications Adapter.
- Recover a specific file from a damaged disk, or recover the entire disk or diskette.
- Print the contents of a graphics display screen on a printer.
- Print files on the printer while the system is doing other work.
- Backup and restore files on a fixed disk.
- Define a remote device as your primary console.

- Sort text data.

- Search files for occurrences of specified strings of text.

- Display a screen full of data at a time.

- Set new system prompt.

- Set the system environment.

- Convert .EXE files to .COM files.

Redirection of Standard Input and Output Devices

The DOS standard input and output device redirection feature allows a program to receive its input from a source other than the (standard input device), or direct its output to a device other than the (standard output device).

DOS provides internal functions that programs can use to receive input and display or print output. These devices are called *standard input devices* and *standard output devices*.

When you first start DOS, the standard input device is the IBM Personal Computer keyboard, and the standard output device is the display unit (screen). However, DOS 2.10 lets you specify input and output devices other than the keyboard and the screen. Refer to the CTTY command, described later on in this chapter, to see how you can specify a remote terminal as the standard input and output device instead of the keyboard and screen.

DOS handles the mapping of the logical (standard input and output) devices to the real devices in a way that is transparent to the application program. The application program does not need to be aware of the physical device that is actually being used for the standard input (STDIN) and standard output (STDOUT) device.

This means that you can run your application program using the keyboard and screen for your input and output device, and then run your application another time using a remote terminal for your input and output device. You do not have to change the application program, which continues to talk to STDIN and STDOUT. DOS goes to the correct physical device on behalf of the application program.

The standard input and output devices can be redirected to or from files or other devices by the following DOS command line parameters:

>[*d*:][*path*]*filename*

> Causes *filename* to be created (or truncated to zero length) and then assigns standard output to that file. All output that would normally have gone to the screen from the command is placed in the file.

>>[*d*:][*path*]*filename*

> Causes *filename* to be opened (created if necessary) and positions the write pointer at the end of the file so that all output is appended to the file.

<[*d*:][*path*]*filename*

> Causes standard input to be assigned to *filename*. All input to the program comes from this file instead of from the keyboard.

CAUTION
When using this method of providing input to a program, be sure *all* of the program's input is in the file. If the program attempts to obtain more input after end-of-file is reached, DOS is unable to supply the input, and processing will stop. You can return to the DOS prompt by entering Ctrl-Break.

Note: If an application does not use DOS function calls to perform standard input and/or output (for example, you put text directly into the video buffer), then redirection will not work for that application.

Example: In this example, the output of the DIR command is sent to the printer:

```
DIR >PRN
```

In this example, the output of the DIR command is sent to file DIRLIST:

```
DIR >DIRLIST
```

In the following example, program MYPROG will receive its input from file INPUT.TXT, instead of from the keyboard:

```
MYPROG <INPUT.TXT
```

Piping of Standard Input and Output

The DOS piping feature allows the standard output of one program to be used as the standard input to another program. DOS uses temporary files to hold the input and output data being piped. These temporary files are created in the root directory of the default drive and have the form:

```
%PIPEx.$$$
```

The programs being piped must use care not to cause the piping files to be erased or modified.

Piping is the chaining of programs with automatic redirection of standard input and output (refer to "Redirection of Standard Input and Output Devices" in this chapter for additional information). The names of the programs to be chained are separated by the vertical bar (¦) character on the command line.

The following are typical examples of using the piping feature for a program that does all of its input and output to the standard input and output devices. For example, if the program named SORT read all of its standard input, sorted it, and then wrote it to the standard output device, the command:

```
DIR¦SORT
```

would generate a sorted directory listing. This causes all standard output generated by the DIR command to be sent to the standard input of the SORT program.

To send the sorted directory to a file, you would type:

```
DIR|SORT>FILE
```

If you wish the file to contain only the directory entries for sub-directories, you could enter:

```
DIR|FIND "DIR"|SORT>FILE
```

DOS Filters

A filter is a program or command that reads data from a standard input device, modifies the data, then writes the result to a standard output device. Thus, the data has been "filtered" by the program. For example, one of the filters on your DOS diskette is called SORT. SORT reads input from the standard input device (normally the keyboard), sorts the lines of data, then writes the sorted results to the standard output device (normally the screen). With the redirection capabilities described earlier in this chapter, you can cause SORT to receive its input from some other source, and to send its output to a different destination. For example,

```
SORT <MYFILE >RESULT
```

will cause SORT to read the file MYFILE, sort the lines within it, and write the sorted output to file RESULT.

By using the piping feature, you can cause a filter to receive its input from the output of another command, or to send its output to the input of another command. For example,

```
DIR|SORT
```

Causes the output listing from the DIR command to be used by SORT as its input. The listing will be sorted and the result displayed on the standard output device.

There are three filters on your DOS diskette, and they are described as individual commands in Chapter 2. They are:

SORT	Sorts text data.
FIND	Searches files for occurrences of specified strings of text.
MORE	Displays a screen full of data at a time, then pauses with the message **—More—**.

You can easily add your own filter to the filters that have been supplied; just write a program that reads its input from the standard input device, and writes its output to the standard output device.

> Note: If an application does not use DOS function calls to perform standard input and/or output (for example you put text directly into the video buffer), then filters will not work for that application.

Invoking a Secondary Command Processor

If you wish to invoke a secondary command processor, the following syntax should be used:

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

Where *d:path* will be the directory searched for the command processor to be loaded, /**P** causes the new copy to become permanent in memory, and /C *string* allows you to pass a command line (string) as a parameter. The command line will be interpreted and acted upon as if you had entered it as a normal command. For example: COMMAND /C DIR B: causes a secondary command processor to be loaded, and it executes the command DIR B:.

Issuing COMMAND without any parameters causes a new copy of the command processor to be loaded, and this new copy will inherit the environment known to the previous level of the command processor. If you use the SET command to change the environment known to the secondary command processor, that change is known *only* to the secondary copy. Exiting back to the primary command processor causes a resumption of the environment that the primary command processor knew before the secondary copy existed. Here's an example. Let's assume that the primary command processor used the normal DOS prompt of A>. If you invoke a secondary copy of the command processor, the secondary copy "inherits" that prompt. If you change the secondary's prompt to something else, and then you exit back to the primary, the primary's prompt will still be A>.

When a secondary command processor has been loaded, you can cause it to return to the previous level of command processor by issuing the special command EXIT. If you use the /P parameter, it will not return to the previous level.

> Note: Application programmers. Please refer to Chapter 7 of IBM *DOS Technical Reference* for additional information.

Types of DOS Commands

There are two types of DOS commands:

- Internal
- External

Internal commands execute immediately because they are built-in to DOS.

External commands reside on disk as program files; therefore, they must be read from disk before they execute. This means that the disk containing the command must already be in a drive, or DOS is unable to find the command. For example, if you entered the command:

```
B:GRAPHICS
```

You must be sure that the diskette containing GRAPHICS.COM is in drive B. If you entered:

```
GRAPHICS
```

Then DOS will look on the default drive (the one in your system prompt) for the GRAPHICS command.

Any file with a filename extension of .COM or .EXE is considered an external command. This allows you to develop your own unique commands and add them to the system. (For example, programs such as FORMAT .COM and COMP.COM are external commands.)

When you select an external command, do not include the filename extension.

For Application Developers: DOS Version 2.00 and 2.10 have provisions that allow you to cause execution of DOS commands from within your application.

> Note: Application developers may find IBM *DOS Technical Reference* helpful. Please refer to Chapter 7 for details.

Format Notation

We will use the following notation to indicate how the DOS commands should be entered:

- You must enter any words shown in capital letters. These words are called *keywords* and must be entered exactly as shown. You can; however, enter keywords in any combination of uppercase and lowercase letters. DOS automatically converts keywords to uppercase.

- You must supply any items shown in lowercase *italic* letters. For example, you should enter the name of *your* file when *filename* is shown in the format.

- Items in square brackets ([ ]) are optional. If you want to include optional information, you do not need to type the brackets, only the information inside the brackets.

- Items separated by a bar (|) mean that you can enter one of the separated items. For example:

  ON | OFF

  Means you can enter **ON** *or* **OFF**, but not both.

- An ellipsis (...) indicates that you can repeat an item as many times as you want.

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

DOS Command Parameters

Parameters are items that you can include in your DOS command statements. They are used to specify additional information to the system. Some parameters are required in your commands, others are optional. If you do not include some parameters, the system provides a default value. Default values that the system provides are discussed in the detailed descriptions of the DOS commands.

Use the following parameters in your DOS command statements:

Parameter	Definition
d:	Denotes when you should specify a drive. Enter a drive letter followed by a colon to specify the drive. For example, A: represents the first drive on your system, B: represents the second. If you omit this parameter, DOS assumes the *default* drive.

Parameter	Definition
path	[\\][*dirname*][\dirname[...]] Denotes a path of directory names. Enter the directory names, separated by backslash characters. If a filename is also to be appended, it should be separated from the last directory name by a backslash. For example: `\DIR1\DIR2\FILE1` The first backslash is optional. If used, it tells DOS to begin with the *root* directory. If omitted, the directory path is assumed to begin with the *current* directory. Global filename characters are not allowed in path specifications. The longest path allowed by DOS (from root directory to the last level) is 63 characters.
filename	Diskette filenames are 1-8 characters in length, and can be followed by a filename extension. The following characters can be used for filenames: A–Z 0–9 $ & # @ ! % " ( ) - { } _ / \\ Any other characters are invalid. An invalid character is assumed to be a delimiter, in which case the filename is truncated. Refer also to "Reserved Device Names" in this chapter for more information about filenames.

Parameter	Definition
.ext	The optional filename extension consists of a period and 1-3 characters. When used, filename extensions immediately follow filenames. The following characters can be used for filename extensions: A–Z 0-9 $ & # @ ! % " ( ) - { } _ / \\ Any other characters are invalid. Remember to include the extension when you refer to a file that has a filename extension; otherwise, DOS will be unable to locate the file.
filespec	[*d*:]*filename*[*.ext*] Examples: **B:myprog.COB** **A:yourprog** **DATAFILE.pas** **cobfile**

Reserved Device Names

Certain names have special meaning to DOS. DOS reserves the following names as system devices:

Reserved Name	Device
CON	Console keyboard/screen. If used as an input device, you can press the F6 key; then press the Enter key to generate an end-of-file indication, which ends CON as an input device.
AUX or COM1	First Asynchronous Communications Adapter port.
COM2	Second Asynchronous Communications Adapter port.
LPT1 or PRN	First Parallel Printer (as an output device only).
LPT2 or LPT3	Second Parallel Printer Third Parallel Printer
NUL	Nonexistent (dummy) device for testing applications. As an input device, immediate end-of-file is generated. As an output device, the write operations are simulated, but no data is actually written.

Notes:

1. Since these are reserved names, you cannot create files with these names.
2. When using a device name, you should assure that the device actually exists; using the name of a nonexistent device can cause unpredictable errors in DOS operation.
3. The reserved device names can be used in place of a filename.
4. Any drive specifier or filename extension entered with these device names will be ignored.

Global Filename Characters

Two special characters ? and * can be used within a filename and its extension. These special characters give you greater flexibility with the DOS commands.

The ? Character

A ? in a filename or in a filename extension indicates that any character can occupy that position. For example,

```
DIR AB?DE.XYZ
```

lists all directory entries on the default drive with filenames that have five characters, begin with AB, have any next character, are followed by DE, and have an extension of XYZ.

Here are some examples of the files that might be listed by the DIR command:

```
ABCDE XYZ
ABIDE XYZ
ABODE XYZ
```

The * Character

An * in a filename or in a filename extension indicates that any character can occupy that position and all the remaining positions in the filename or extension. For example,

```
DIR AB*.XYZ
```

lists all directory entries on the default drive with filenames that begin with AB and have an extension of XYZ. In this case, the filenames may be from 2-8 characters in length.

Here are some example files that might be listed by the DIR command:

```
ABCDE   XYZ
ABC357  XYZ
ABIDE   XYZ
ABIIOU  XYZ
ABO$$$  XYZ
AB      XYZ
```

Examples of Ways to Use ? and *

Example 1

To list the directory entries for all files named INPUT on drive A (regardless of their filename extension), enter:

```
DIR A:INPUT.???
  or
DIR A:INPUT.*
```

Example 2

To list the directory entries for all files on drive A (regardless of their filenames) with a filename extension of XYZ, enter:

```
DIR A:????????.XYZ
  or
DIR A:*.XYZ
```

Example 3

To list the directory entries for all files on drive A with filenames beginning with ABC and extensions beginning with E, enter:

```
DIR A:ABC?????.E??
  or
DIR A:ABC*.E*
```

Information Common to all DOS Commands

This section gives a description of how to use the commands presented in Chapter 2. The commands appear in Chapter 2 in alphabetical order; each with its purpose, format, and type. Examples are provided where appropriate.

The following information applies to all DOS commands:

- The normal prompt from the command processor is the default drive letter plus >, such as A> unless changed by the PROMPT command.
- When a command completes, the system prompt will reappear on the screen. If no error messages are displayed before the system prompt reappears, the command has been successfully completed.
- Commands are usually followed by one or more parameters.
- Commands and parameters may be entered in uppercase or lowercase, or a combination of both.
- DOS will search the current directory of the specified or default drive to find a command or batch file whose name you have entered. If not found, DOS will continue its search in each of the directories listed in the PATH command.

- Most commands allowing you to enter filenames will also accept a path (directory) name ahead of the filename. If you do not plan to create directories of your own, you may disregard all references to path names. This will greatly simplify the command syntax for you.

- Commands and parameters *must* be separated by delimiters (space, comma, semicolon, equal sign, or the tab key). The delimiters can be different within one command. For example, you could enter:

```
COPY oldfile.rel;newfile.rel
RENAME,thisfile thatfile
```

- The three parts of filespec (*d:filename.ext*) must not be separated by delimiters. The colon (:) and period (.) already serve as delimiters.

- In this book, we usually use a space as the delimiter in the commands for readability.

- Also in this book, when we say *"Press any key"*, we mean "Press any *character* key."

- Files are not required to have filename extensions when you create or rename them; however, you must include the filename extension when referring to a file that *has* a filename extension.

- You can end commands while they are running by pressing Ctrl-Break. Ctrl-Break is recognized only while the system is reading from the keyboard or printing characters on the screen, unless you have used BREAK=ON in your configuration file or have issued a BREAK=ON command. Thus, the command may not end immediately when you press Ctrl-Break.

- Commands become effective only after you press the Enter key.

- Global filename characters and device names are not allowed in a command name. You may only use them in command parameters.

- For commands displaying a large amount of output, you can press Ctrl-Num Lock to suspend the display of the output. You can then press any character key to continue the display.

- You can use the control keys and the DOS editing keys described in *DOS User's Guide* while entering DOS commands.

- Drives will be referred to as *source* drives and *target* drives. A source drive is the drive you will be transferring information *from*. A target drive is the drive you will be transferring information *to*.

- When an external command is entered, DOS first looks for it in the current directory of the default or specified drive. If not found, DOS continues searching for it in the directories listed in the most recent PATH command.

- If the characters <, >, or ¦ appear anywhere in the command line you enter, DOS will act upon them as described in "Redirection of Standard Input and Output" and "Piping of Standard Input and Output Device" described in this chapter.

Thus, the command:

```
REM this is a | test
```

would pipe the output of the REM command (none) to a program named "test". If the program "test" does not exist, this message appears:

```
Bad command or filename
```

DOS Version 2.10 Features

DOS Versions 2.00 and 2.10 contain several features for more efficient programming. These features include automatic program execution, using batch files, and enhancements from prior versions of DOS. For more information on DOS enhancements refer to Appendix E. DOS Versions 2.00 and 2.10 contain the same functions.

By using automatic program execution you may start a specific program every time you start DOS. The concept of automatic program execution is explained on the following page.

A batch file is a file containing one or more commands that DOS executes one at a time. Batch files allow you to set up a group of commands or procedures that automatically go into effect when the batch file is executed. For more information on batch files refer to "Batch Commands" in Chapter 2.

We recommend that you take the time to review the information in Appendix E, whether you are an experienced DOS user or are about to use DOS for the first time.

Automatic Program Execution

You may want to start a specific program every time you start DOS. You can do this with the DOS command processor by using *automatic program execution*.

Every time you start up DOS, the command processor searches for a file named AUTOEXEC.BAT in the root directory on the disk that DOS was started from. This filename is special because it refers to a *batch file* that is automatically executed whenever you start the system. With this facility, you can execute programs or commands immediately every time you start DOS.

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

If DOS does not find the AUTOEXEC.BAT file, DOS issues the date and time prompts. Refer to "Batch Commands" in Chapter 2 for details on how to create an AUTOEXEC.BAT file.

Chapter 2. DOS Commands

Contents

Introduction

This chapter presents a detailed description of the DOS commands. The commands appear in alphabetical order. The description includes the purpose, format, and type of each command. Examples are provided where appropriate.

Note: If you have not yet done so, we suggest you read "Chapter 1. Using DOS" before continuing.

ASSIGN (DRIVE) Command

Purpose: Instructs DOS to use a different drive from the one that was specified for disk operations.

Format: ASSIGN [*x*=*y* [...]]

Type: Internal External
***

Remarks: Use this command to tell DOS to route all requests for a disk drive to a different drive.

The first drive letter *x* is internally converted by DOS to the second drive letter *y*. This command does not require you to enter a colon after the drive letter. Entering ASSIGN with no parameters causes all drive reassignments to be reset so that normal drive assignments will resume.

Example: This example causes DOS to route all requests for drive A to drive C. Thus, if you issue DIR A: DOS will display the directory that is on physical drive C:

```
ASSIGN A=C
```

In this example, any requests for drive A or drive B are routed by DOS to drive C:

```
ASSIGN A=C B=C
```

The command:

```
ASSIGN
```

will *undo* the reassignment so that requests for drive A will again go to physical drive A, etc.

> **Note:** This command has been included to assist you with applications that were designed to perform their disk operations specifically on drives A and B (those applications that do not allow you to specify a drive). By using a command such as:
>
> ```
> ASSIGN A=C B=C
> ```
>
> Those applications can be made to use drives other than A and B, such as a fixed disk.

Reassignment of drives should *only* be used when necessary for these cases. It should never be used with the PRINT command or when running DOS in normal operations, because it can *hide* the true device type from commands and programs that require actual drive information. Also note that DISKCOPY and DISKCOMP will ignore any drive reassignments.

If you will be developing an application program we recommend that you avoid using specific drive assignments within your program. Instead, allow the user to specify the drive(s) to be used.

BACKUP (Fixed Disk) Command

Purpose:	Backs up one or more files from a fixed disk to diskettes.	
Format:	BACKUP *d*:[*path*][*filename*[.*ext*]] *d*:[/S][/M][/A][/D:*mm-dd-yy*]	
Type:	Internal	External ***

Remarks: Use DOS formatted diskettes only. The first parameter you specify is the fixed disk file you want to back up. The second parameter is the backup diskette drive. Files are backed up from the current directory if you do not specify a path. If you do not specify a filename or extension, then all files in the directory will be backed up.

Global filename characters are allowed in the filename. They cause all files matching the filename to be backed up onto diskettes. For example, entering:

```
BACKUP C:*.DAT A:
```

Causes each file from the current directory of fixed disk drive C that has an extension of .DAT to be backed up onto the diskette in drive A.

The parameter /S causes the files in all sub-directories to be backed up in addition to the files in the specified directory itself. This includes sub-directories at all levels beyond the specified directory.

The parameter /M indicates that only files that have been modified since the last backup should be backed up. Use this parameter to avoid backing up files that never change. The BACKUP command can tell which files have been changed because of an indicator in each file's directory entry that is set by DOS whenever the file is written to.

The parameter /A indicates that backed up files should be added to the files on the backup diskette already in the specified drive. If this parameter is omitted, then you will be warned that all previously existing files on the diskette will be erased.

The parameter /D can be used to back up files written only on or after the specified date. See the description of the DATE command for valid date formats.

The following example backs up all of the files on fixed disk drive C:

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

The next example backs up three different files from the default fixed disk drive onto the same set of backup diskettes:

```
BACKUP \level1\file1.dat A:
BACKUP \level1\level2\file2.dat A: /A
BACKUP \level1\level3\file3.dat A: /A
```

The next example backs up all files in the current directory that have changed since the last backup:

```
BACKUP *.* A: /M
```

After you enter the BACKUP command, you will be prompted to insert a diskette (unless you specified the /A parameter). Use DOS formatted diskettes only. BACKUP will erase existing files on the diskette before it starts backing up the fixed disk file, unless you have used the /A parameter. After BACKUP fills up a diskette, it will prompt you to insert a new diskette. You should label each diskette and record the date and diskette number.

BACKUP displays the name of each file as it backs it up. If you want a printed copy of this list, you can use redirection of output to the printer. Refer to "Redirection of Standard Input and Output Devices" in Chapter 1 for additional information.

The BACKUP command sets the exit code as follows:

0	Normal completion
1	No files were found to backup
3	Terminated by user (Ctrl-Break)
4	Terminated due to error

These codes can be used with the batch processing IF subcommand.

Note: The files on the backup diskettes are unusable in normal processing, and should only be used by the RESTORE command.

Batch Commands

Purpose: Executes the commands contained in the specified file from the current directory of the designated or default drive. If the batch file is not found in the current directory, DOS searches for it in the directories listed in the PATH command.

Format: [*d*:]*filename* [*parameters*]

Type:

Internal	External
***	

Remarks: A batch file is a file containing one or more commands that DOS executes one at a time. All batch files must have a filename extension of .BAT.

You can pass parameters to the *filename*.BAT file when the file executes. Therefore, the file can do similar work with different data during each execution.

You create a batch file by using the Line Editor (EDLIN), or by using the COPY command directly from the standard input device.

Notes:

1. Do not enter the name BATCH (unless the name of the file you want to execute is BATCH.BAT).

2. Only the filename must be entered to run the batch file. Do not enter an extension.

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

4. There are seven subcommands that can be used to control batch processing: ECHO, FOR, GOTO, IF, SHIFT, PAUSE, and REM. They are explained in the following pages.

5. If you press Ctrl-Break while in batch mode, this prompt appears:

   ```
   Terminate 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.

6. If you remove a diskette containing a batch file being processed, DOS prompts you to insert it again before the next command can be read.

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

8. DOS will *remember* which directory your batch file was started from. Therefore, the commands within the batch file may change the current directory at will, and the batch file will continue executing.

The AUTOEXEC.BAT File

The AUTOEXEC.BAT file is a special batch file. When you start or restart DOS, the command processor searches for the AUTOEXEC.BAT file. If this file is present in the root directory of the drive DOS was started from, DOS automatically executes the file whenever you start DOS.

For example, if you want to automatically load BASIC and run a program called MENU, create an AUTOEXEC.BAT file as follows:

1. Enter:

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

   This statement tells DOS to copy the information from the standard input device into the AUTOEXEC.BAT file on the default drive.

2. Now, enter:

```
BASIC MENU
```

   and press Enter.

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

3. Press the F6 key, then press the Enter key to end copying from the standard input device to the file.

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

To run your own BASIC program, enter the name of your program in place of MENU in the second line of the example. Remember, you can enter any DOS command, or series of commands, in the AUTOEXEC.BAT file.

Note: If you use AUTOEXEC.BAT, DOS does not prompt you for the current date and time unless you include DATE and TIME commands in the AUTOEXEC.BAT file.

Creating a .BAT File With Replaceable Parameters

Within a batch file you may include *dummy* parameters that can be replaced by values supplied when the batch file executes.

For example, enter:

```
A>Copy con: ASMFILE.BAT
Copy %1.MAC %2.MAC
Type %2.PRN
Type %0.BAT
```

Press Enter after entering each line.

Now, press F6; then press Enter.

The system responds with this message:

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

The file ASMFILE.BAT, which consists of three commands, now resides on the diskette in the default drive.

The dummy parameters %0, %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.

Notes:

1. Up to 10 dummy parameters (%0 – %9) can be specified within a batch file, more than 10 parameters can be specified on a command line (see SHIFT subcommand).

2. If you want to use % as part of a filename *within* a batch file, you must specify it twice. For example, to specify the file ABC%.EXE you must enter it as ABC%%.EXE in the batch file.

Executing a .BAT File With Replaceable Parameters

To execute the ASMFILE.BAT file and pass parameters, enter the batch filename followed by the parameters you want sequentially substituted for %1, %2, etc.

For example, you can enter:

```
ASMFILE A:PROG1 B:PROG2
```

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

The result is the same as if you entered each of the three commands (in the ASMFILE.BAT file) from the console with their parameters, as follows:

```
Copy A:PROG1.MAC B:PROG2.MAC
Type B:PROG2.PRN
Type ASMFILE.BAT
```

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

ECHO Subcommand

Purpose: The ECHO batch processing subcommand allows or inhibits the screen display of DOS commands executed from a batch file. It does not interfere with messages produced while the commands are executing.

Format: ECHO [ON|OFF|*message*]

Type:	Internal	External
	***	

Remarks: Batch commands are normally displayed on the screen as they are read from the batch file. ECHO is ON after power-on or system reset. ECHO ON displays all the commands on the standard output device as they are executed. ECHO OFF stops the display of commands on the screen (including the REM command).

Echo *message* displays message on the standard output device regardless of the current ON or OFF state. In this way, you can cause specific messages to be displayed even when ECHO has been turned off. If ECHO is issued with no parameters, the current ECHO state (ON or OFF) is displayed.

Example: In this example, the batch file contains the following:

```
echo  off
rem  ****  command display is off
dir  a:/w
echo on
dir  a:/w
```

Upon execution of the above batch file, the following display will occur:

```
echo off

Volume on drive A has no ID
Directory of A:\

filename1.ext  filename2.ext

2 file(s)    xxxxx bytes free

dir  a:/w

Volume in drive A has no ID
Directory of A:\

filename1.ext    filename2.ext

2 file(s)    xxxxx bytes free
```

In the above example, the ECHO OFF is displayed. The rem command and dir a:/w are not displayed because ECHO is OFF, but the output of the dir is not inhibited.

Batch Commands

FOR Subcommand

Purpose: The FOR batch processing subcommand allows iterative execution of DOS commands.

Format: FOR %%*variable* IN (*set*) DO *command*

Type:

Internal	External
***	

Remarks: The *%%variable* is sequentially set to each member of *set* and then the *command* is evaluated and executed. If a member of *set* is an expression involving * and/or ?, then the *%%variable* is set to each matching filename from disk.

Example: In this example, if you enter the command:

```
FOR %%f IN (prog1.asm prog2.asm prog3.asm) DO dir
%%f
```

The result is

```
dir prog1.asm
dir prog2.asm
dir prog3.asm
```

Note: FOR subcommands cannot be nested; that is, only one FOR subcommand can be specified on a command line. Also, path names are not accepted with filenames.

GOTO Subcommand

Purpose: The GOTO batch processing subcommand transfers control to the line following the one containing the appropriate label. A label is inserted in a batch file as a colon (:) followed by the label name.

Format: GOTO *label*

Type:	Internal	External
	***	

Remarks: The GOTO *label* causes commands to be executed beginning with the line immediately after *:label*. If *:label* is not defined, the current batch file terminates with the message **Label not found.** A label in a batch file is defined as a character string where the first 8 characters are significant (make it different).

Example: In this example, the following batch file produces an indefinite sequence of **rem looping...** and **GOTO LOOP** messages on the screen:

```
:LOOP
rem looping...
GOTO LOOP
```

Note that labels within a batch file are never displayed while the batch file is executing. In the example above, the line **:LOOP** would not be displayed. Thus, unreferenced labels provide a handy means for placing comments within your batch file that are not displayed when the file is executed.

IF Subcommand

Purpose:	The IF batch processing subcommand allows conditional execution of DOS commands.	
Format:	IF [NOT] *condition command*	
Type:	Internal ***	External

Remarks: The *condition* parameter is one of the following:

ERRORLEVEL *number*

string1==string2

EXIST*filespec*

When the **IF** parameter's condition is true, then the DOS command is executed. Otherwise, the DOS command is skipped, and the following command in the file is executed.

ERRORLEVEL *number* is true if the previous program had an exit code of *number* or higher. The number is specified as a decimal value.

String1==string2 is true when *string1* and *string2* are identical.

> Note: The corresponding characters of String1 and String2 must both be uppercase or lowercase to be identical.

EXIST*filespec* is true if *filespec* is found on the specified drive. Path names are not allowed with the *filespec*.

NOT *condition* is true if the *condition* is false.

Example: This example is for the IF EXIST*filespec* command:

```
if exist file1 goto abc
          .
          .
          .
:abc
command
```

Execution of a batch file containing this command would make the condition true provided *file1* is found on the default drive. The *goto abc* would be executed causing the system to skip to the command following the label *:abc*. If *file1* is not found, the *goto abc* would not be executed. Processing would then continue with the next command in the batch file.

The following example is for the IF string1==string2 command:

```
if %1 == Doug echo Doug was here!
```

Execution of a batch file containing this command with *Doug* given as the %1 parameter would make the condition true. The ECHO batch command would then be executed displaying **Doug was here!**. If *Doug* was not given as the %1 parameter, the condition would have been false. The ECHO batch command would not have been executed. Processing would continue with the next command in the batch file.

The following example is for the IF ERRORLEVEL *number* command:

```
myprog1
if errorlevel 1 echo myprog1 failed – end batch file execution
```

The above two commands are in a batch file; MYPROG1 is a program that sets the errorlevel when it completes its processing. In the simple case, MYPROG1 sets the errorlevel to 0 if it completed processing successfully and sets errorlevel to 1 if processing completed unsuccessfully. The batch file conditional **if errorlevel 1 echo ...** tests for the situation when MYPROG1 failed. If MYPROG1 completed processing unsuccessfully, the condition is true and the ECHO batch command is executed. The

ECHO batch command displays the data (or message) immediately following *echo*. If MYPROG1 was successful, the condition would not be true and the ECHO batch command would not be executed. Processing would then continue with the next command in the batch file.

The following example is for the IF NOT EXIST *filespec* command:

```
if not exist a:%1 copy b:%1 a:
program
```

The above IF batch command demonstrates the NOT condition. The batch file that this command is in, is going to execute a program that requires a particular file to be on drive A. The IF command is executed prior to the program to make sure that the required file is not on drive A. If the file does not exist on drive A, the condition is true. The *copy* will then be executed, copying the file from drive B to drive A to satisfy the requirements of the program. If the file does exist on drive A, the copy will not be executed. Processing will then continue to execute the batch file.

> Note: At the present time, only BACKUP and RESTORE commands set an ERRORLEVEL that can be tested. The facility is included to allow your own programs to set an error code that can then be interrogated by the IF ERRORLEVEL subcommand.

PAUSE Subcommand

Purpose: Suspends system processing and displays the message **Strike a key when ready....**

Format: PAUSE [*remark*]

Type:	Internal	External
	***	

Remarks: You can insert PAUSE commands within a batch file to display messages and to give you the opportunity to change diskettes between commands. To resume execution of the batch file, press any key *except* Ctrl-Break. (Ctrl-Break ends processing.)

If you include the optional *remark*, the remark is also displayed. The optional remark can be any string of characters up to 121 bytes long.

You can control how much of a batch file you want to execute by placing PAUSE commands at strategic points in the file. At each PAUSE command, the system stops and gives you time to decide whether to end processing. To end processing, press Ctrl-Break. To continue processing, press any other key.

Example: If you enter this PAUSE command in a batch file, the following message is displayed:

```
A>PAUSE Change diskette in drive A
Strike a key when ready... _
```

This PAUSE enables you to change diskettes between commands.

REM (Remark) Subcommand

Purpose: Displays remarks from within a batch file.

Format: REM [*remark*]

Type: Internal *** External

Remarks: The remarks are displayed when the batch execution reaches the remark.

Remarks can be any string of characters up to 123 bytes long.

You can use REM commands without remarks for spacing within your batch file, for readability.

Example: If the following REM command is issued in a batch file, this remark is displayed:

```
REM This is the daily checkout program
```

Note: See "Information Common to All DOS Commands" in this Chapter 1 for considerations in using the REM command.

SHIFT Subcommand

Purpose: The SHIFT batch processing subcommand allows command lines to make use of more than 10 (%0 through %9) replaceable parameters.

Format: SHIFT

Type:	Internal	External
	***	

Remarks: Replaceable parameters are numbered %0 through %9. If you wish to use more than 10 parameters on a command line, you can use SHIFT to get at parameters past the tenth. All parameters on the command line are shifted one position to the left, with the %0 parameter being replaced by the %1 parameter, etc. Each subsequent shift command causes all the parameters to be shifted to the left by one position. For example:

```
%0 = A
%1 = B
%2 = C
%3 = D
 .
 .
 .
%9
```

The SHIFT results are:

```
%0 = B
%1 = C
%2 = D
 .
 .
 .
%9
```

Example: This example demonstrates how the SHIFT subcommand can be used in a batch file. If a batch file named MYFILE.BAT contains the following commands, and the default drive is A:

```
echo %0 %1 %2 %3
shift
echo %0 %1 %2 %3
shift
echo %0 %1 %2 %3
shift
echo %0 %1 %2 %3
shift
echo %0 %1 %2 %3
```

Invoke the batch file with the following parameters:

```
MYFILE PROG1  PROG2  PROG3
```

Produces the following results:

```
A>echo MYFILE  PROG1  PROG2  PROG3
MYFILE  PROG1  PROG2  PROG3

A>shift

A>echo  PROG1  PROG2  PROG3
PROG1  PROG2  PROG3

A>shift

A>echo PROG2  PROG3
PROG2  PROG3

A>shift

A>echo PROG3
PROG3
A>shift
A>echo

A>
```

BREAK (Control Break) Command

Purpose:	Allows you to instruct DOS to check for a control break whenever a program requests DOS to perform any functions (such as disk operations).	
Format:	BREAK [ON | OFF]	
Type:	Internal ***	External

Remarks: Use this command to specify when DOS should check for a Ctrl-Break being entered at the keyboard. Normally, DOS only performs this check during standard input, standard output, standard print device, or auxiliary device operations.

Ctrl-Break allows you to *breakout* of a program that performs few or no standard device operations (such as a compiler).

The **ON** parameter causes DOS to begin checking for Ctrl-Break *whenever* a program requests any DOS function. Specifying **OFF** causes DOS to check only during standard input, standard output, standard print device, or Asynchronous Communications Adapter operations.

You can also turn on the extended checking by using BREAK=ON in your configuration file. Refer to "Configuring Your System" in Chapter 4.

Entering **BREAK** with no parameter causes DOS to display the current state (on or off) of Ctrl-Break checking.

CHDIR (Change Directory) Command

Purpose: Change the DOS current directory of the specified or default drive, or to display the current directory path of a drive.

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

or

CD [[*d*:]*path*]

Type:	Internal	External
	***	

Remarks: The current directory is where DOS looks to find files whose names were entered without specifying which directory they were in. If you do not specify a drive, the default drive is assumed. If you enter **CHDIR** or **CD** with no parameters or with only a *d*: parameter, the current directory path of the specified or default drive is displayed.

Example: In this example, the command will change the current directory of the default drive to its root directory:

```
CHDIR \
```

In this example, the command will change drive B's current directory to the path "root——>LEVEL1——>LEVEL2":

```
  CD  B:\LEVEL1\LEVEL2
```

In this example, the command will change drive B's directory to the current directory path plus LEVEL3:

```
  CD  B:LEVEL3
```

Thus, if the second example had been used, the resultant path would be:

```
  root——>LEVEL1——>LEVEL2——>LEVEL3
```

The search for the LEVEL3 directory begins in the directory that was current when the command was issued, because no leading backslash (\) was used.

If the command is entered as:

```
CD  B:\LEVEL3
```

then DOS looks for the path:

```
root——>LEVEL3
```

because the leading backslash (\) tells DOS to start at the root directory.

DOS remembers the current directory for each drive on the system and, if you don't specify a path, any reference to a drive will access the current directory.

Example:

```
B>CD \LEVEL1\LEVEL2

B>CD A:\LEVEL1

B>COPY B:JACK.COM A:
```

The file JACK.COM is copied from the LEVEL1\LEVEL2 directory on drive B to the \LEVEL1 directory on drive A.

CHKDSK (Check Disk) Command

Purpose:	Analyzes the directories and the File Allocation Table on the designated or default drive and produces a disk and memory status report.	
Format:	CHKDSK [*d*:][*filename*[*.ext*]][/F][/V]	
Type:	Internal	External ***

Remarks: If you specify a filename, CHKDSK will display the number of non-contiguous areas occupied by the file or files. Note that CHKDSK only looks in the current directory for these files.

CHKDSK will not automatically correct errors found in the directory or file allocation table. If you want it to make the corrections, use the /F (fix) parameter. If you do *not* specify the /F parameter, CHKDSK continues to function as though it were preparing to correct the disk so that you can analyze the possible results of correction, but does not actually write the corrections on the disk.

If you use the /V parameter, CHKDSK will display a series of messages indicating its progress, and provide more detailed information about the errors it finds.

After checking the disk, CHKDSK displays any error messages, followed by a status report. A complete listing of error messages can be found in Appendix A.

Following is an example of the status report that is displayed:

```
Volume MYDISK       Created AUG 12, 1983  10:00

   179712 bytes total disk space
    18944 bytes in 3 hidden files
      512 bytes in 1 directories
    26112 bytes in 4 user files
   134144 bytes available on disk

   196608 bytes total memory
   170736 bytes free
```

Note that in this status report, *three hidden* files were reported. These are the volume label, and the DOS system files IBMBIO.COM and IBMDOS.COM, that are hidden from the normal directory searches. Some application programs also create hidden files.

CHKDSK does not wait for you to insert a diskette. It assumes that the diskette to be checked is in the specified drive. Therefore, on a single diskette-drive system, it is especially important that the specified drive is different from the default drive, unless you are checking the DOS diskette itself.

You should run CHKDSK occasionally for each fixed disk and diskette to ensure the integrity of the file structures.

Notes:

1. All yes or no (Y/N) prompts from CHKDSK require you to press Enter after entering Y or N, to prevent accidental changes to your disk.

2. If you specified a filename, the number of non-contiguous areas occupied by the file will be reported. Badly fragmented files (many non-contiguous areas) can cause system performance to slow down when those files are accessed, since DOS can not read them sequentially. You can determine the extent of file fragmentation by using *.* in the filename field of the CHKDSK command.

3. If CHKDSK finds *lost* allocation units (clusters) on the disk, it asks if you wish to recover the lost data into files. If you say yes, and the /F parameter was used, CHKDSK recovers each chain of lost allocation units into a file whose name is in the form:

   FILE*nnnn*.CHK

Where *nnnn* is a sequential number starting with 0000. These files are created in the root directory of the specified drive. You can then look at these files to see if they have any useful information. If not, you can erase them.

4. If you redirect CHKDSK's output to a file, for example:

   ```
   CHKDSK B:>FILE
   ```

   it will report errors on that file. In this case, be sure not to use the /F parameter.

CLS (Clear Screen) Command

Purpose:	Clears the display screen.	
Format:	CLS	
Type:	Internal ***	External

Remarks: This command clears the display on the standard output device upon execution. If foreground and background colors have been selected through the "Extended Screen Control and Keyboard" functions described in Chapter 2 of the *Technical Reference* manual, the colors will remain unchanged. Otherwise, the standard output device is set to white characters on a black background.

COMP (Compare Files) Command

Purpose: Compares the contents of the first set of specified files to the contents of the second set of specified files. Usually, you would run COMP after a COPY operation to ensure that the two sets of files are identical.

> **Note:** This command compares two sets of *files*; the DISKCOMP command compares two *entire diskettes*.

Format: COMP [*d*:][*path*][*filename*[*.ext*]] [*d*:][*path*][*filename*[*.ext*]]

Type:	Internal	External
		***

Remarks: The first parameter you specify is the *primary* file. The second parameter is the *secondary* file. The files that you compare may be on the same drive or on different drives. They can also be in the same directory or different directories.

Global filename characters are allowed in both filenames, and will cause all of the files matching the first filename to be compared with the corresponding files from the second filename. Thus, entering:

```
COMP A:*.ASM B:*.BAK
```

causes each file that has an extension of .ASM from drive A to be compared with a file of the same name (but with an extension of .BAK) from drive B.

If no parameters are entered with the COMP command, or if the second parameter is missing, you are prompted for them. If either parameter only contains a drive or a path with no filename, COMP assumes a filename of *.*. You can enter a complete path with either of the two filenames. For example, the command:

```
COMP B:*.ASM C:
```

causes all .ASM files on drive B to be compared with the files of the same names and extensions on drive C.

It is also possible to compare all files in one directory with all corresponding files in another directory. For example:

```
COMP A:\LEVEL1 A:\LEVEL2
```

locates all the files in the LEVEL1 directory of drive A and compares them with files of the same name from the LEVEL2 directory on the same drive.

If no file matches the primary filename, COMP will prompt you for both the primary and secondary parameters again.

The paths and names of the files being compared are displayed as the comparing process proceeds. An error message will follow if a file matching the second filespec cannot be found, or the files are different sizes, or a specified directory path is invalid.

During the comparison, an error message appears for any location that contains mismatching information in the two files. The message indicates the offset into the files of the mismatching bytes, and the contents of the bytes themselves (all in hexadecimal), as follows:

```
Compare error at offset XXXXXXXX
File 1 = XX
File 2 = XX
```

In this example, File 1 is the first filename entered; File 2 is the second filename entered.

After ten unequal comparisons, COMP concludes that further comparing would be useless; processing ends; and the following message is displayed:

```
10 Mismatches – ending compare
```

After a successful comparison, COMP displays:

```
Files compare OK
```

After the comparison of the two files ends, comparing will proceed with the next pair of files that match the two filenames, until no more files can be found that match the first parameter. Then COMP displays:

```
Compare more files (Y/N)? _
```

You now have the option to compare two more files or to end the comparison. If you want to compare two more files, enter **Y**. You will be prompted for new primary and secondary filenames.

If you want to end COMP processing, enter **N**. You will return to the DOS prompt. In all compares, COMP looks at the last byte of the files being compared to assure that it contains a valid end-of-file mark (Ctrl-Z, which is the hexadecimal character 1A). If found, no action is taken by COMP. If the end-of-file mark is *not* found, COMP produces the message:

```
EOF mark not found
```

This is done because some products produce files whose sizes are always recorded in the directory as a multiple of 128 bytes, even though the actual usable data in the file is usually a few bytes less than the directory size. In this case, COMP may produce **compare error** messages when comparing the few bytes beyond the last real data byte in the last block of 128 bytes (COMP always compares the number of bytes reflected in the directory). Thus, the **EOF mark not found** message indicates that the compare errors may not have occurred in the usable data portion of the file.

Notes:

1. The two sets of files you want to compare can have the same path and filenames—provided they are on different drives.

2. If you only specify a drive for the second file, it is assumed that the second filename is the same as the first filename. But, the current directory is to be used. That is, it is the same as entering:

   d:*.*

3. A comparison does not take place if the file sizes are different.

4. COMP does not wait for you to insert a diskette containing a file to be compared. Therefore, if a file to be compared is not on the same diskette as the COMP command itself, you should enter COMP with no parameters. When COMP prompts for the filenames, you can insert the desired diskette and reply with the name of the file to be compared.

COPY Command

Purpose: Copies one or more files to another disk and optionally, gives the copy a different name if you specify it in the COPY command.

COPY also copies files to the same disk. In this case, you *must* give the copies different names unless different directories are specified; otherwise, the copy is not permitted. Concatenation (combining of files) can be performed during the copying process.

You can also use COPY to transfer data between any of the system devices. (An example of how to copy information that you enter at the keyboard to a file is provided at the end of the description of COPY.)

Format: COPY [/A][/B][*d*:][*path*]*filename*[.*ext*][/A][/B] [*d*:][*path*][*filename*[.*ext*]][/A][/B][/V]

or

COPY [/A][/B][*d*:][*path*]*filename*[.*ext*][/A][/B] [+[*d*:][*path*]*filename*[.*ext*][/A][/B]...] [*d*:][*path*][*filename*[.*ext*]][/A][/B][/V]

Type:

Internal	External
***	

Remarks: The first file specified is the source file. The second file specified is the target file. If the second parameter is a directory (*path* with no filename), files are copied into that directory without changing their names.

The parameter /V causes DOS to verify that the sectors written on the target diskette are recorded properly. Although errors in recording data are very rare, this option has been provided for those of you who wish to verify that critical data has been correctly recorded. This option causes the COPY command to run more slowly, due to the additional overhead of verification.

The /V parameter provides the same check as does the VERIFY ON command. /V is redundant if the VERIFY ON command has been executed previously. The difference is that /V is effective only during the duration of the COPY command. The VERIFY ON command is in effect continually until VERIFY OFF is entered.

The parameters /A and /B indicate the amount of data to be processed by the COPY command. Each applies to the filespec preceding it and to all remaining filespecs on the command line until another /A or /B is encountered. These parameters have the following meanings:

When used with a *source* filespec:

/A Causes the file to be treated as an ASCII (text) file. The file's data is copied up to, but not including, the first end-of-file character (Ctrl-Z, which is hex 1A) found in the file; the remainder of the file is not copied.

/B Causes the entire file (based on the directory file size) to be copied.

When used with a *target* filespec:

/A Causes a Ctrl-Z character to be added as the last character of the file.

/B Causes no end-of-file character (Ctrl-Z) to be added.

The default values are /A when concatenation is being performed (see Option 3 below), and /B when concatenation is not being performed (Options 1 and 2).

Notes:

1. When copying to or from a reserved device name, the copy is performed in ASCII (/A) mode. The first Ctrl-Z character encountered will end the copy unless /B was specified.

2. When making a copy of a file that is marked read-only, the copy will not be marked read-only.

You can use the global characters ? and * in the filename and in the extension parameters of both the original and duplicate files. If you enter a ? or * in the source *filespec*, the names of the files will be displayed as the files are being copied. For more information about global characters, refer to “Global Filename Characters” in Chapter 1.

COPY Command

The COPY command has three format options:

Option 1 – Copy With Same Name

Use this option to copy a file with the copied file having the *same* filename and extension as the source file. For example:

COPY [*d*:][*path*\]*filename*[.*ext*]

or

COPY [*d*:][*path*\]*filename*[.*ext*] *d*:[*path*]

In the first example, we want to copy a file to the current directory of the default drive. In the second example, we specify the target drive and/or directory. In both examples, because we did not specify the second filename, the copied file will have the same filename as the source file. Because we did not specify a name for the second file, the source drive and the target drive must be different unless different directories were specified or implied; otherwise, the copy is not permitted.

For example, assume the default drive is A. The command:

```
COPY B:MYPROG
```

copies the file MYPROG from drive B, to the default drive A, with no change in the filename. The command:

```
COPY *.* B:
```

copies all the files from the default drive A to drive B, with no change in the filenames or in the extensions. The filenames are displayed as the files are copied. This method is very useful if the files on drive A are fragmented. The command:

```
COPY B:\MYPROG   B:\LEVEL1
```

copies the file MYPROG from the root directory of drive B to the directory path:

```
root——>LEVEL1
```

on the same drive. The copy has the same filename as the original file. Note that the above example assumes that directory \LEVEL1 exists on drive B. If it did not, then the file MYPROG would have been copied into a file named LEVEL1 in the root directory of drive B. In other words, if the second parameter specifies a directory that exists, the file (or files) will be placed in that directory, keeping the same filename. If the second parameter does not specify a directory that exists, DOS will treat it as a filename.

Option 2 – Copy With Different Name

Use this option when you want the copied file to have a different name from the file that is being copied. For example:

COPY [*d*:][*path*]*filename*[*.ext*] [*path*]*filename*[*.ext*]

or

COPY [*d*:][*path*]*filename*[*.ext*]
d:[*path*]*filename*[*.ext*]

In the first example, we copied a file (first file specified), and renamed the copy (second file specified). We did not specify a drive, so the default drive was used. In the second example, we copied a file and renamed the copy also. In this example, we did specify the target drive. Because we changed the name of the file, the source drive and the target drive do not have to be different. The current directory can be the same or different.

For example:

```
COPY MYPROG.ABC B:*.XXX
```

copies the file MYPROG.ABC from the diskette in the default drive to drive B, naming the copy MYPROG.XXX. The current directory of each drive was used.

You can also use reserved device names for the copy operation. For example:

```
COPY CON fileA
COPY CON AUX
COPY CON LPT1
COPY fileA CON
COPY fileB AUX
COPY fileC LPT2
COPY AUX LPT1
COPY AUX CON
```

Also, NUL: can be used in any variation.

Refer to "Reserved Device Names" in Chapter 1 for information about system devices.

This example shows how to use COPY to put what you enter from the keyboard into a file:

```
A>COPY CON fileA
Type a line and press Enter.
Type your next line and press Enter.
•
•
•
Type your last line and press Enter.
Now, press F6 and then press Enter.
```

When you press F6, and then press Enter, the COPY operation ends and saves the information you entered. In this example, the information is saved in a file named fileA.

> **Note:** This example assumes that you have not altered the meaning of F6 through the "Extended Keyboard Support" functions described in Chapter 2 of the IBM *DOS Technical Reference* manual. If you have, then substitute the key that you have assigned Ctrl-Z for F6 in this example.

Option 3 – Copy and Combine Files

Use this option when you want to combine files while copying. That is, you can combine two or more files into one file by adding the additional files to the end of the first. The date and time recorded in the result file directory are the current date and time. The message indicating the number of files copied refers to the number of result files created.

To combine files, list any number of source files, separated by plus (+) signs in the COPY command. Use the following format:

COPY [/A][/B][*d*:][*path*]*filename*[.*ext*][/A][/B]

[+[*d*:][*path*]*filename*[.*ext*][/A][/B]...]

[*d*:][*path*]*filename*[.*ext*]][/A][/B][/V]

For example:

```
COPY A.XYZ+B.ABC+B:C.TXT BIGFILE.TXT
```

This command creates a new file called **BIGFILE.TXT** on the default drive. The combination of **A.XYZ**, **B.ABC**, and **B:C.TXT** is put into **BIGFILE.TXT**.

If you do not specify a result *filename*, the additional files are added to the end of the first file, leaving the result in the first file. For example:

```
COPY A.ASM+B.ASM
```

In this case, COPY appends **B.ASM** to the end of **A.ASM** and leaves the result in **A.ASM**.

> Note: Combining files is normally performed in text (or ASCII) mode. That is, the first Ctrl-Z (hex 1A) character in the file is interpreted as an end-of-file mark. To combine binary files, use the **/B** parameter to force COPY to use the physical end-of-file (the file length shown in the DIR command).

You can also combine ASCII and binary files by using the following parameters:

- ASCII – /A
- Binary – /B

For example:

```
COPY A.XYZ+B.COM/B+B:C.TXT/A BIGFILE.TXT
```

An /A or /B takes effect on the file it is placed after, and it applies to all subsequent files on the command line until another /A or /B is found. An /A or /B on the result file causes a Ctrl-Z to be added (/A), or not to be added (/B), as the last character in the result file.

You can use the global characters ? and * in the filenames of both the files to be combined and the result file. For example:

```
COPY *.LST COMBIN.PRN
```

In this example, all files matching *.LST are combined into one file called COMBIN.PRN::

```
COPY *.LST+*.REF COMBIN.PRN
```

This example combines all files matching *.LST and then all files matching *.REF into one file called COMBIN.PRN:

```
COPY *.LST+*.REF *.PRN
```

In this example, each file matching *.LST is combined with the corresponding .REF file, with the result having the same name but with extension .PRN. Thus, a file FILE1.LST would be combined with FILE1.REF to form FILE1.PRN; XYZ.LST would be combined with XYZ.REF to form XYZ.PRN; etc. Note that in this case (when multiple files are to be created), only one file from each of the source filespecs is used to create a given target file.

For more information about global characters, refer to "Global Filename Characters" in Chapter 1.

It is easy to enter a COPY command to combine files where one of the source files is the same as the target, yet this often cannot be detected. For example:

```
COPY *.LST ALL.LST
```

This would produce an error if ALL.LST already existed. The error would not be detected, however, until it was time for ALL.LST to be appended; by this time, ALL.LST could already have been altered.

COPY handles this situation as follows: As each input file is found, its name is compared with the target filename. If the names are the same, that one input file is skipped, and the message **Content of destination lost before copy** is displayed. Further copying proceeds normally. This allows *summing* files with a command like:

```
COPY ALL.LST + *.LST
```

This command appends all .LST files, except ALL.LST itself, to ALL.LST. In this case, the error message is suppressed, because this is a true *physical append* to ALL.LST.

The following are special cases. Remember to use the **/B** parameter whenever you use the plus (+) sign with non-ASCII files:

```
COPY B:XYZ.ASM+
```

This command copies the file XYZ.ASM to the default drive and gives it a new date and time. To simply change the date and time, leaving the file in place, you can use the following command:

```
COPY B:XYZ.ASM+,, B:
```

Note that the two commas are necessary to define the end of the *source filename*, because COPY normally expects to see another filename after the plus (+) sign.

In these special cases, if global filename characters are used in the filename or extension, then all of the matching files will be appended together into the first filename that matches. Thus, the command:

```
COPY B:*.*+,, B:
```

will not update the dates and times of all files on drive B, but will append all of drive B's files together into a single file that will replace the first file found on drive B.

> **Note:** When combining files, COPY considers the copying process to be successful if at least one, but not necessarily all, of the named source files is found. If none of the source files can be found, you receive the message:
>
> ```
> 0 file(s) copied
> ```

CTTY (Change Console) Command

Purpose: Changes the standard input and output console to an auxiliary console, or restores the keyboard and screen as the standard input and output devices.

Format: CTTY *device–name*

Type:	Internal	External
	***	

Remarks: Defines the device to be used as the primary console. Specifying AUX, COM1, or COM2 causes DOS to use that device as the primary console. Specifying CON resets the standard input and output device to the primary console.

Example: In this example, the command causes DOS to use the AUX device for its standard input and output operations:

```
CTTY AUX
```

In this example, the command reverses the previous assignment, causing DOS to switch back to the standard screen and keyboard for its operations:

```
CTTY CON
```

Notes:

1. The CTTY command accepts the name of *any* character-oriented device to allow you to install your own device drivers, and to specify their device names. You must be certain that the named device is capable of both input and output operations. For example, you should not specify the name of a printer, because DOS will attempt to read from that device.

2. The CTTY command is effective only for programs that use DOS function calls. Other programs, such as BASIC (that do not use DOS function calls), will not be able to use the CTTY command to change the standard input and output devices.

DATE Command

Purpose: Permits you to enter a date or change the date known to the system. The date is recorded in the directory entry for any files you create or alter.

Format: DATE [*mm-dd-yy*]

Type:	Internal	External
	***	

Remarks: If you enter a valid date with the DATE command, the new date is accepted, and the system prompt appears. Otherwise, the DATE command issues the following prompt:

```
Current date is day mm-dd-yy
Enter new date:_
```

The system displays the day of the week in the *day* location.

Enter a new date in the form *mm–dd–yy* or *mm/dd/yy,* where:

mm	is a one- or two-digit number from 1-12
dd	is a one- or two-digit number from 1-31
yy	is a two-digit number from 80-99 (a leading 19 is assumed) or a four-digit number from 1980-2099

You can change the date from the standard input device or from a batch file. Remember, when you start the system, you are not prompted for the date if you use an AUTOEXEC.BAT file. You may want to include a DATE command in that file. For more information about the AUTOEXEC.BAT file, refer to "Batch Commands" in this chapter.

Notes:

1. To leave the date as is, press Enter.
2. The valid delimiters within the date are hyphens (-) and slashes (/).
3. Any date is acceptable as today's date, as long as the digits are in the correct ranges.
4. If you enter an invalid date or delimiter, you receive an **Invalid date** message.

Example: In this example, once you press Enter, the date known to the system is 7/24/83.

```
A>DATE
Current date is Mon 1-18-1983
Enter new date: 7/24/83
```

DEL Command

See “ERASE Command” in this chapter.

DIR (Directory) Command

Purpose: Lists either all the directory entries, or only those for specified files. The information provided in the display includes the volume identification and the amount of free space left on the disk. The display line for each file includes its size in decimal bytes and the date and time the file was last written to. Entries that name other directories are clearly identified with <DIR> in the file size field.

> **Note:** Directory entries for system files IBMBIO.COM and IBMDOS.COM are not listed, even if present.

Format: DIR [*d*:][*path*][*filename*][*.ext*][/P][/W]

Type:	Internal	External
	***	

Remarks: The **/P** parameter causes the display to pause when the screen is full. When you are ready to continue with the directory listing, press any key.

The /W parameter produces a wide display of the directory, which lists only the filenames and directory names. Each line displayed contains five names. (This parameter is only recommended for 80-column displays.)

You can use the global characters ? and * in the filename and extension parameters. For more information about the global characters, refer to "Global Filename Characters" in this Chapter 1.

The DIR command has two format options (the **/P** and **/W** parameters may be used with either option):

Option 1 – List All Files

Use this option to list all the files in a directory. For example:

DIR [*path*]

or

DIR *d*:[*path*]

In the first example, we want to list all directory entries on the default drive. In the second example, we want to list all directory entries on the specified drive. In both cases, if a path is specified, the listing is of files in the specified directory. Otherwise, the current directory is listed.

The directory listing might look like this:

```
A>dir

  Volume in drive A is MYDISK
  Directory of A:\

FILE1      A         10368        7-20-83    12:13p
FILE3      A          1613        5-27-83    12:14p
9X                      31        8-17-82    10:59a
LEVEL2          <DIR>             9-09-82    12:10p
FILE1                 2288        9-02-82    12:25p
              5 File(s) 141312 bytes free
```

Note that if the directory being listed is not the root directory, the above example would have included two special entries. The first entry would contain a period in place of a filename. The second would contain two periods in place of a filename. The list of files shown above would follow these two entries. These special entries tell you that the directory being listed is a subdirectory, rather than the root directory.

Option 2 – List Selected Files

Use this option to list selected files from a directory. For example:

DIR [*path*]*filename.ext*

or

DIR *d*:[*path*]*filename.ext*

If either *filename* or *.ext* is omitted, an * is assumed.

In the first example, we want to list all the files in the named or current directory of the *default* drive that have the specified filename and extension. In the second example, we want to list all the files in the named or current directory of the *specified* drive that have the specified filename and extension.

Using the previous example, if you enter:

```
dir file3.a
```

the screen might look like this:

```
A>dir file3.a

 Volume in drive A is MYDISK
 Directory of  A:\

FILE3     A      1613      5-27-83     12:14p
          1  File(s)  141312 bytes free
```

If you enter:

```
dir *.a
```

or

`dir .a` (omission of *filename* defaults to *)

the screen might look like this:

```
A>dir *.a

Volume in drive A is MYDISK
Directory of A:\

FILE1    A        10368        7-20-83    12:13p
FILE3    A         1613        5-27-83    12:14p
         2 File(s)  141312 bytes free
```

If you enter:

```
dir file1
```

the screen might look like this (omission of *.ext* defaults to *):

```
  A>dir file1
   Volume in drive A is MYDISK
   Directory of A:\

FILE1      A        10368        7-20-83    12:13p
FILE1                2288        9-02-82    12:25p
           2 File(s)  141312 bytes free
```

To display only the entry for a file that has no extension, enter the filename followed by a period. In this case, the *.ext* does *not* default to *. For example,

```
dir file1
```

displays the entry for FILE1, but not for FILE1.A.

If you wish to display all the files in directory LEVEL2 on the above drive, you can enter:

```
dir level2
```

The screen will look like this:

```
A>dir level2

Volume in drive A is MYDISK
Directory of A:\level2

 .            <DIR>          9-09-82
 ..           <DIR>          9-09-82
MYPROG    COM       2463     7-30-82   8:55a
          3 File(s)  141312 bytes free
```

Note that all files in directory LEVEL2 have been listed, including the two special entries found in all sub-directories. The entry marked with a single period denotes the directory being listed (LEVEL2), and the double period denotes this directory's parent directory (in this case, the root directory). Thus, if your *current* directory is LEVEL2 and you wish to see the files in its parent directory, you can enter:

```
dir ..
```

The following screen is displayed:

```
  A>dir ..

  Volume in drive A is MYDISK
  Directory of A:\

FILE1      A                  10368   7-20-83    12:13p
FILE3      A                   1613   5-27-83    12:14p
9X                               31   8-17-82    10:59a
LEVEL2          <DIR>                 9-09-82    12:10p
FILE1                          2288   9-02-82    12:25p
          5 File(s)  141312 bytes free
```

DISKCOMP (Compare Diskette) Command

Purpose: Compares the contents of the diskette in the first specified drive to the contents of the diskette in the second specified drive. Usually, you would run DISKCOMP after a DISKCOPY operation to ensure that the two diskettes are identical.

Notes:

1. This command is used only for comparing diskettes. If a fixed disk drive letter is specified, an error message is displayed.

2. This command compares two *entire diskettes*; the COMP command compares two *files*.

Format: DISKCOMP [*d:*] [*d:*] [/1] [/8]

Type:

Internal	External
	***

Remarks: You can specify the same drive or different drives in this command. If you specify the same drive, a single-drive comparison is performed. You are prompted to insert the diskettes at the appropriate time. DISKCOMP waits for you to press any key before it continues.

The /1 parameter forces DISKCOMP to compare only the first side of the diskettes, even if the diskettes and drives are dual-sided.

The /8 parameter causes DISKCOMP to compare only 8 sectors per track, even if the first diskette contains 9 sectors per track.

DISKCOMP compares all tracks on a track-for-track basis and issues a message if the tracks are not equal. The message indicates the track number and the side (0 or 1) where the mismatch was found.

After completing the comparison, DISKCOMP prompts:

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

If you press Y, the next comparison is done on the same drives that you originally specified, after you receive prompts to insert the proper diskettes.

To end the command, press N.

Notes:

1. If you omit both parameters, a single-drive comparison is performed on the default drive.

2. If you omit the second parameter, the default drive is used as the secondary drive. If you specify the default drive in the first parameter, this also results in a single-drive comparison.

3. On a single-drive system, all prompts are for drive A, regardless of any drive specifiers entered.

4. DISKCOMP usually does not issue a **Diskettes compare OK** message if you try to compare a backup diskette created by the COPY command with the diskette you copied from. The COPY operation produces a copy that contains the same information, but may place the information at different locations on the target diskette from those locations used on the source diskette. In this case, you should use the COMP command to compare individual files on the diskettes.

5. If a diskette error occurs while DISKCOMP is reading the diskette, a message is produced that indicates where (track and side) the error occurred. Then DISKCOMP continues to compare the rest of the diskette. Because the remainder of the data to be compared cannot be read correctly from the indicated track and side, you can expect to receive a **Compare error** message.

6. DISKCOMP automatically determines the number of sides and sectors per track to be compared, based on the diskette that is to be read first (the first drive parameter entered).

   If the first diskette or drive can be read on only one side, or if the /1 parameter is used, only the first side is read from both diskettes. If the first diskette contains 9 sectors per track, then DISKCOMP will compare 9 sectors per track unless you used the /8 parameter. If the first drive and diskette are dual-sided, and /1 is not specified, a two-sided comparison is done. In this case an error message is produced if either the second drive or the diskette is a single-sided diskette.

DISKCOPY (Copy Diskette) Command

Purpose: Copies the contents of the diskette in the source drive to the diskette in the target drive. The target diskette is formatted if necessary, during the copy.

> **Note:** This command is used only for copying diskettes. If a fixed disk drive letter is specified, an error message is displayed.

Format: DISKCOPY [*d:*] [*d:*] [/1]

Type:	Internal	External
		***

Remarks: The first parameter you specify is the source drive. The second parameter is the target drive.

The /1 parameter causes DISKCOPY to copy only the first side of the diskette, regardless of the diskette or drive type.

You can specify the same drives or different drives. If the drives are the same, a single-drive copy operation is performed. You are prompted to insert the diskettes at the appropriate times. DISKCOPY waits for you to press any key before continuing.

After copying, DISKCOPY prompts:

```
Copy another (Y/N)?_
```

If you press Y, the next copy is done on the same drives that you originally specified, after you are prompted to insert the proper diskettes.

To end the command, press N.

Notes:

1. If the target diskette has not been formatted with the same number of sides and sectors per track as the source diskette, DISKCOPY will format the target diskette during the copy operation.

2. If you omit both drive parameters, a single-drive copy operation is performed on the default drive.

3. If you omit the second parameter, the default drive is used as the target drive.

4. If you omit the second parameter and you specify the default drive as the source drive, a single-drive copy operation is performed.

5. On a single-drive system, all prompts will be for drive A, regardless of any drive letter you may enter.

6. Diskettes that have had a lot of file creation and deletion activity become fragmented, because diskette space is not allocated sequentially. The first free sector found is the next sector allocated, regardless of its location on the diskette.

   A fragmented diskette can cause degraded performance due to excessive head movement and rotational delays involved in finding, reading, or writing a file. If this is the case, we recommended that you use the COPY command, instead of DISKCOPY, to eliminate fragmentation.

   For example, place a freshly formatted diskette in drive B, and the diskette you wish to copy in drive A. The command:

   ```
   COPY A:*.* B:
   ```

   copies all the files from the diskette in drive A to the diskette in drive B. The resultant files (in drive B) are now copied sequentially. You should get better performance when you use these files from now on.

7. You can run DISKCOMP after a successful DISKCOPY to ensure that the diskettes are identical.

8. If disk errors are encountered on either diskette, DISKCOPY indicates the drive, track, and side in error and proceeds with the copy. In this case, the target diskette (copy) may or may not be usable, depending on whether the affected diskette location was to contain valid data.

9. DISKCOPY automatically determines the number of sides and sectors per track to copy, based on the source drive and diskette. If only the first side of the source diskette can be read, then only the first side can be copied. If the source drive and diskette are dual-sided, both sides can be copied (unless you override it with the /1 parameter). In this case, if the target drive is single-sided, an error message will indicate that the drives are incompatible.

   If the source diskette has ever been physically formatted with 9 sectors per track, then all 9 sectors on each track will be copied.

ERASE Command

Purpose: Deletes the file with the specified filename from the specified directory on the designated drive, or deletes the file from the default drive if no drive is specified. If no path is entered, the file is deleted from the current directory.

Format: ERASE [*d*:][*path*][*filename*[.*ext*]]

or

DEL [*d*:][*path*][*filename*[.*ext*]]

Type:	Internal	External
	***	

Remarks: The shortened form, DEL, is a valid abbreviation for ERASE.

You can use the global characters ? and * in the filename and in the extension. Global characters should be used with caution, however, because multiple files can be erased with a single command. For more information about global characters, refer to "Global Filename Characters" in Chapter 1.

To erase all files in the current directory, enter:

```
ERASE [d:]*.*
```

To erase all files in a specific directory, enter:

```
ERASE [d:]path
```

ERASE assumes a filename of *.* if no filename is given.

Notes:

1. The system files IBMBIO.COM and IBMDOS.COM cannot be erased. Also, the two special entries in each subdirectory (. and .. in place of filenames) cannot be erased.

2. If you use the filespec *.* to erase all of the files in a directory or on a disk, DOS issues the following message to verify that you actually want to erase all files:

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

   If you *do* want to erase all of the files on the diskette, enter **Y**. Otherwise, enter **N**. Then press the Enter key.

Example: In this example, the file **myprog.1** will be erased from the current directory of drive A.

```
A>ERASE A:myprog.1
```

EXE2BIN Command

Purpose: Converts .EXE files that have no segment fixup to a form that is compatible with .COM programs. This results in a saving of diskette space and faster program loading.

Format: EXE2BIN [*d*:][*path*]*filename*[*.ext*]] [*d*:][*path*][*filename*[*.ext*]]

Type:

Internal	External
	***

Remarks: 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, [*d*:]*filename*[*.ext*]. If you do not specify a drive, the drive of the input file is used. If you do not specify an output filename, the input filename is used. If you do not specify a filename extension in the output filename, the new file is given an extension of .BIN. If you do not specify a path, the current directory is used.

The input must be in valid .EXE format as 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 the specified initial CS:IP:

- If CS:IP is not specified in the program (the .EXE file contains 0:0), a pure binary conversion is assumed. If segment fixups are necessary (the program contains instructions requiring segment relocation), you are prompted for the fixup value. This value is the absolute segment at which the program is to be loaded.

  In this case, the resultant program is usable only when loaded at the absolute memory address specified by a user application. The DOS command processor will not be capable of properly loading the program.

- 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; that is, they must assume the entry conditions explained in Chapters 1, 4, 5, 6, 7, 8, and 9 of the IBM *DOS Technical Reference* manual and Appendixes B, C, and D of this manual. Once the conversion is complete, you may rename the resultant file to a .COM extension. Then, the command processor is capable of loading and executing the program in the same manner as the .COM programs supplied on your DOS diskette.

If CS:IP does not meet one of these criteria, or if it meets the COM file criterion but has segment fixups, the following message is displayed:

```
File cannot be converted
```

This message is also displayed if the file is not a valid .EXE file.

To produce standard COM files with the assembler, you must both use the assembler statement ORG to set the location pointer of the file at 100H and specify the first location as the start address. (This is done in the END statement.) Also, the program must not use references that are defined only in other programs. For example, with the IBM Personal Computer MACRO Assembler:

```
ORG 100H
START:
 •
 •
 •
END START
```

EXE2BIN resides on your DOS Supplemental Program diskette.

FDISK Command

See Chapter 3, ‘‘Preparing Your Fixed Disk’’.

FIND Filter Command

Purpose:	This filter sends to the standard output device all lines from the filenames specified in the command line that contain the specified string.
Format:	FIND [/V][/C][/N]*string*[[*d*:][*path*]*filename*[.*ext*]...]
Type:	Internal External ***

Remarks: The /V parameter causes all lines *not* containing the *string* to be displayed.

The /C parameter causes FIND to display only a count of the number of matching occurrences of *string* in each file, without displaying the matching lines from the file.

The /N parameter causes the relative line number of each matching line to be displayed ahead of the line from the file.

The string should be enclosed in double quotes. Two quotes in succession are taken as a single quote.

Global filename characters are not allowed in the filenames or extensions.

Example:
```
A>FIND "Fool's Paradise" book1.txt book2.txt book3
```

will output all lines from the book1.txt, book2.txt, and book3 (in that order) that contain the string "Fool's Paradise". Or,

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

will output the names of all the files in the current directory of drive B that do not contain the string DAT.

FORMAT Command

Purpose: Initializes the disk in the designated or default drive to a recording format acceptable to DOS; analyzes the entire disk for any defective tracks; and prepares the disk to accept DOS files by initializing the directory, File Allocation Table, and system loader.

CAUTION
Please note that formatting destroys all data on the disk. Because of this you should be very careful before you decide to format any disk, particularly a fixed disk. If you attempt to format your fixed disk, please note that the entire contents of any previously created DOS partition, including all subdirectories and their contents, are destroyed. If you are not certain which drive is the default drive, do NOT enter FORMAT without a drive letter. If you do not specify a drive letter, you could unintentionally destroy all of the data on whatever disk happens to be default. It's safer to specify a drive letter.

Format: FORMAT [*d*:][/S][/1][/8][/V][/B]

Type:	Internal	External
		***

Remarks: You must format all new diskettes (by using either the FORMAT or DISKCOPY command) and fixed disks (through FORMAT) before they can be used by DOS.

A fixed disk must also be formatted again if you change the size of its DOS partition through the FDISK command.

If you specify /S in the FORMAT command, the operating system files are also copied from the default drive to the new disk or diskette in the following order:

```
IBMBIO.COM
IBMDOS.COM
COMMAND.COM
```

If you specify /1, the target diskette is formatted for single sided use, regardless of the drive type.

If you specify /8, the target diskette is formatted for use at 8 sectors per track. FORMAT will default to 9 sectors per track usage if you do not specify /8. Note that FORMAT always creates 9 physical sectors on each diskette track, but that it instructs DOS to use only 8 sectors per track if you use the /8 parameter. The /1 and /8 parameters are valid only for diskettes.

If you specify /V, FORMAT will prompt you for a volume label which will be written on the disk. We strongly recommend that you use the /V parameter. This will uniquely identify each disk.

FORMAT does not allow you to enter /8 and /V at the same time. This is because DOS will not recognize a volume label on 8 sectored diskettes. This maintains compatibility with DOS 1.10.

The volume label cannot be used in place of filenames as input to any of the DOS commands. The volume label is for your use in keeping track of your diskettes.

The /B parameter causes FORMAT to create an 8 sector per track diskette with space allocated for the IBMBIO.COM and IBMDOS.COM system modules. It does not place the system modules or the command processor on the diskette. This parameter is used to create a diskette on which any version of DOS (1.00, 1.10, 2.00, or 2.10) can be placed through that version's SYS command. If the /B parameter is not used, only DOS Version 2.00 or 2.10 can be placed on the diskette through the SYS command.

The /V and /S parameters cannot be used with the /B parameter.

Notes:

1. Formatting destroys any previously existing data on the disk.

2. During the formatting process, any defective tracks are marked as *reserved* to prevent the tracks from being allocated to a data file.

3. Directory entries for IBMBIO.COM and IBMDOS.COM are marked as *hidden files*, and therefore, they do not appear in any directory searches—including the DIR command.

4. FORMAT will prompt you to enter a volume label (volume identification) if you have used the /V parameter. The label can consist of from 1 to 11 characters. All characters acceptable in filenames are acceptable in the volume label. Unlike filenames, however, the volume label does not contain a period between the eighth and ninth characters.

5. FORMAT produces a status report, that indicates:

   - Total disk space
   - Space marked as defective
   - Space currently allocated to the DOS system files (when /S is used)
   - Amount of space available for your files

6. FORMAT determines the target drive type and formats the disk or diskette accordingly. For diskettes, if the diskette can be successfully read and written on only one side, the diskette is formatted for single-sided use; it can be used in either type of drive. If the target drive is dual-sided and you do not use the /1 parameter, the diskette is formatted for dual-sided use; it will not be usable in a single-sided drive.

7. Fixed disks are already physically formatted (proper recording format) when shipped by IBM. When formatting a fixed disk, FORMAT checks all locations within the DOS partition, but does not physically format them again.

8. If the /S parameter is used and the system has insufficient available memory for FORMAT to load all three system modules, it will load as many modules as it can, format the target disk and write the modules that are in memory. It must then read the remaining modules from the source disk so they can be placed on the target disk. If the source diskette has been removed from the drive, an appropriate message will prompt you to reinsert it before FORMAT can continue.

Example: By issuing the following command, the diskette in drive B is formatted and the operating system files are also copied:

```
A>FORMAT B:/S/V
```

The system displays the following message:

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

After you insert the appropriate diskette and press any key, the system displays this message:

```
Formatting...
```

while the diskette formatting is taking place.

Once the formatting is complete, the system displays this message:

```
Formatting...Format complete
System transferred

Volume label (11 character, ENTER for none)? MYDISK

  xxxxxx bytes total disk space
   xxxxx bytes used by system
  xxxxxx bytes available on disk

Format another (Y/N)?n
```

In the above example, note that MYDISK was entered as the volume label.

Enter Y to format another diskette.

Enter N to end the FORMAT program.

When you format a fixed disk, you will see the following message instead of the prompt to insert a diskette:

```
Press any key to begin formatting x:
```

The *x* is replaced by the drive letter you typed. Other than this, the messages appear in the same manner.

Fixed disk formatting can take several minutes because of the large size that is allocated to DOS, so don't be alarmed if it takes some time before you are prompted for the volume label. You can tell that FORMAT is working by noting that your fixed disk drive light is on.

GRAPHICS (Screen Print) Command

Purpose:	Allows the contents of a graphics display screen to be printed on an IBM Personal Computer 80 cps Graphics Printer when using a color/graphics monitor adapter. This command increases the resident size of DOS in memory by 736 bytes.

Format: GRAPHICS

Type:	Internal	External
		***

Remarks: Press the Shift-PrtSc keys to print the screen contents on the printer. If the screen is in text mode, the text is printed in under 30 seconds. If the screen is in the graphics mode, each time the PrtSc key is pressed, the following things occur:

- In the 320x200 color graphics mode (Screen 1), the screen contents are printed in up to four shades of gray.

- In the 640x200 color graphics mode (Screen 2), the screen is printed sideways on the paper. The upper right corner of the screen is printed on the upper left corner of the paper.

- Printing may take as long as three minutes.
- To invoke the screen print from a program, use the following coding example:

```
PUSH BP
INT 5
POP BP
```

MKDIR (Make Directory) Command

Purpose: Creates a subdirectory on the specified disk.

Format: MKDIR [*d*:]*path*

or

MD [*d*:]*path*

Type:	Internal	External
	***	

Remarks: If you do not specify a drive, the default drive is assumed.

Example: In this example, the command creates an entry in the root directory for a new subdirectory called LEVEL2:

```
MD \LEVEL2
```

If you have done this and wish to add another directory level, you can use either of the following two examples.

Use this example if your current directory is the root directory:

```
MD \LEVEL2\LEVEL3
```

adds an entry for subdirectory LEVEL3 in the LEVEL2 directory.

Use this example if your current directory is LEVEL2:

```
MD LEVEL3
```

This command will do the same thing as the previous example. Note that in the previous example, the first backslash (\) tells DOS to begin its directory search with the *root* directory. The absence of a leading \ in the last example causes DOS to begin at the *current* directory. Each directory may contain the names of still other directories.

> **Note:** You can create as many subdirectories as you wish, limited only by available disk space. However, you should ensure that the maximum length of any single path from the root directory to the desired level is 63 characters, including imbedded backslashes.

MODE Command

Purpose: Sets the mode of operation on a printer or on a display connected to the Color/Graphics Monitor Adapter, sets options for an Asynchronous Communications Adapter, or causes printer output to be routed to an Asynchronous Communications Adapter.

> **Technical Note:** When used in Option 1, 3, or 4, the MODE command causes printer and Asynchronous Communications Adapter intercept code to be made resident in memory. This increases the size of DOS in memory by approximately 256 bytes.

Format: MODE LPT#:[*n*][,[*m*][,P]]

or

MODE *n*

or

MODE [*n*],*m*[,T]

or

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

or

MODE LPT#:=COM*n*

MODE Command

Type:	Internal	External
		***

Remarks: A missing or invalid *n* or *m* parameter means that the mode of operation for that parameter is not changed. The MODE command has four format options:

Option 1 (For the printer)

MODE LPT#:[*n*][,[*m*][,P]]

where:

#	is 1, 2, or 3 (the printer number)
n	is 80 or 132 (characters per line)
m	is 6 or 8 (lines per inch vertical spacing)
P	specifies continuous retry on time-out errors

For example:

```
MODE LPT1:132,8
```

sets the mode of operation of printer 1 to 132 characters per line and 8 lines per inch vertical spacing. The power-on default options for the printer are 80 characters per line and 6 lines per inch.

The retry loop can be stopped by pressing Ctrl-Break. To stop time-out errors from being continuously retried when you have entered **P**, you must use MODE Option 1 without specifying **P**.

Option 2 **(For switching Display Adapters, and setting the display mode of the Color/ Graphics Monitor Adapter)**

MODE *n*

or

MODE [*n*],*m*[,T]

where:

n	is 40, 80, BW40, BW80, CO40, CO80, or MONO
40	sets the display width to 40 characters per line (for Color/Graphics Monitor Adapter).
80	sets the display width to 80 characters per line (for Color/Graphics Monitor Adapter).
BW40	switches the active display adapter to the Color/Graphics Monitor Adapter, and sets the display mode to Black and White (disables color) with 40 characters per line.
BW80	switches the active display adapter to the Color/Graphics Monitor Adapter, and sets the display mode to Black and White (disables color) with 80 characters per line.

CO40	switches the active display adapter to the Color/Graphics Monitor Adapter, enables color, and sets the display width to 40 characters per line.
CO80	switches the active display adapter to the Color/Graphics Monitor Adapter, enables color, and sets the display width to 80 characters per line.
MONO	switches the active display adapter to the Monochrome Display Adapter (which always has display width of 80 characters per line).
m	is R or L (shift display right or left).
T	requests a test pattern used to align the display.

For readability, you can shift the display one character (for 40 columns) or two characters (for 80 columns) in either direction. If you specify T in the MODE command, a prompt asks you if the screen is aligned properly. If you enter Y the command ends. If you enter N the shift is repeated followed by the same prompt. For example,

```
MODE 80,R,T
```

sets the mode of operation to 80 characters per line and shifts the display two character positions to the right. The test pattern is displayed to give you the opportunity to further shift the display without having to enter the command again.

Option 3 (For Asynchronous Communications Adapter)

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

where:

n	Either 1 or 2 (Asynchronous Communications Adapter number)
baud	110, 150, 300, 600, 1200, 2400, 4800, or 9600 **Note:** Only the first two characters are required; subsequent characters are ignored.
parity	Either **N** (none), **O** (odd), or **E** (even) – (default = E)
databits	Either **7** or **8** (default = 7)
stopbits	Either **1** or **2** (if baud equals 110, default = 2; if baud does not equal 110, default = 1)

These are the *protocol* parameters. They are used to initialize the Asynchronous Communications Adapter. When you specify the protocol, you must specify at least the baud rate. The other parameters can be omitted, with the defaults accepted, by entering only commas. For example:

```
MODE COM1:12,N,8,1,P
```

sets the mode of operation to 1200 baud rate, no parity, eight databits, and one stopbit. To use the defaults listed in the definitions above, you enter:

```
MODE COM1:12,,,,P
```

The *parity* defaults to even, the *databits* defaults to seven, and the *stopbits* defaults to one.

The **P** option indicates that the asynchronous adapter is being used for a serial interface printer. If you enter the **P**, time-out errors are continuously retried. You can stop the retry loop by pressing Ctrl-Break. To stop the time-out errors from being continuously retried when you have entered **P**, you must reinitialize the asynchronous adapter without entering the **P**.

Option 4 (To redirect parallel printer output to an Asynchronous Communications Adapter)

```
MODE LPT#:=COMn
```

where:

#	Either 1, 2, or 3 (printer number)
n	Either 1 or 2 (Asynchronous Communications Adapter number)

All output directed to printer LPT# is redirected to the Asynchronous Adapter *n*.

Notes:

1. Before you can use MODE to redirect parallel printer output to a serial device, you must initialize the Asynchronous Communications Adapter by using Option 3 (see above). If that serial device is a printer, your serial initialization command should also include the **P** parameter.

2. MODE LPT#:[*n*][,*m*] disables the redirection for the printer designated by the #.

MORE Filter Command

Purpose: This filter reads data from the standard input device, and sends one screen full of data to the standard output device, and then pauses with the message ——More——.

Format: MORE

Type: Internal External
***

Remarks: Pressing any character key causes another screen full of data to be written to the standard output device. This process continues until all input data is read.

Example: In this example, the command line will display the contents of file TEST.ASM one screen full at a time. When the screen is full, the message —More— appears on the bottom line. You can press any key to see the next screen full.

```
MORE <TEST.ASM
```

PATH (Set Search Directory) Command

Purpose: Causes specified directories to be searched for commands or batch files that were not found by a search of the current directory.

Format: PATH [[*d:*]*path*[[;[*d:*]*path*]...]]

Type:

Internal	External
***	

Remarks: You may specify a list of drives and path names, separated by semicolons (note that path names must be specified and will not default to current directory). Then, when you enter a command that is not found in the current directory of the drive that was specified (or implied) with the command, DOS searches the named directories in the sequence you entered them. The current directory is not changed.

Entering PATH with no parameters causes DOS to display the names that were specified on a previous PATH command (that is, the search paths currently defined to DOS). Entering PATH with only a semicolon (PATH;) resets the search path to null (no extended search path). This is the default when DOS is started. In this case, DOS searches only the current directory for commands and batch files.

Example: In this example, assume the program MYPROG.COM resides only in directory MYDIR on drive B, and that the default drive is drive A:

```
PATH \LEVEL;\LEVEL2\LEVEL3;B:\MYDIR
```

This command instructs DOS to look in the current directory of the drive specified, followed by A:\LEVEL2, then A:\LEVEL2\LEVEL3 then B:\MYDIR until it finds the command you have entered. If the command entered is *not* found in any of the directories specified in PATH, the message **Bad command or filename** is displayed.

In the previous example, if you enter the command:

```
MYPROG
```

DOS searches four directories, finding the program MYPROG in B:\MYDIR.

Notes:

1. Erroneous information in the paths, such as invalid drive specifications or imbedded delimiters, will not be detected until DOS actually needs to search the specified paths.

2. If a path is specified that no longer exists at the time DOS uses it to search for a command or batch file, DOS ignores that path and goes on to the next.

PRINT Command

Purpose: Prints a queue (list) of data files on the printer while you are doing other tasks on the computer. Up to 10 filenames can be queued for printing at one time. The first time this command is issued, it increases the resident size of DOS in memory by approximately 3200 bytes.

Format: PRINT [[*d*:][*filename*[.*ext*]][/T][/C] [/P]...]

Type:	Internal	External
		***

Remarks: You can enter more than one filename on the command line, each with appropriate parameters. Global filename characters * and ? are allowed in the filename and extension. **Only files in the current directory can be queued for printing.** Once a file has been queued, you can change the current directory without affecting the printing of the files already in the print queue.

/T sets the terminate mode. All queued files are canceled from the print queue. If a file is currently being printed, the printing stops, a cancellation message is printed, the paper is advanced to the next page, and the printer's alarm sounds.

/C sets the cancel mode. Allows you to select which file or files to cancel. The preceding filename and all following filenames entered on the command line are canceled from the print queue until a /**P** is found on the command line, or the Enter key is pressed.

/**P** sets the print mode. The preceding filename and all following filenames are added to the print queue until a /C is found on the command line, or the Enter key is pressed.

Global filename characters * and ? are allowed in the filename and extension. You can enter more than one filename on the command line, each with appropriate parameters.

If no / parameters are specified and the Enter key is pressed, the files listed on the command line are queued for printing (/**P** is assumed).

If PRINT is entered with no parameters, PRINT displays the names of the files currently in the print queue.

The first time the PRINT command is executed after you start your system, the following message is displayed on the display screen:

```
Name of list device [PRN]:
```

This allows you to specify the output list device, LPT1, LPT2, LPT3, PRN, COM1, COM2, AUX, etc. The default is PRN, and will be selected if you press Enter.

> **Note:** Be sure the device you name is physically attached to your system; naming a nonexistent device will cause unpredictable system behavior.

The files are queued for printing in the order entered. After each file is printed, the printer paper is advanced to the next page. Any tab characters found are expanded with blanks to the next 8-column boundary.

If PRINT encounters a disk error while attempting to read the file to be printed, PRINT will cause:

- The file currently printing to be canceled
- The disk error message to be printed on the printer
- The printer paper to be advanced to the next page and the alarm to be sounded
- The remaining files in the print queue to be printed

If the /T or /C parameters are used to cancel a file or files currently being printed:

- A file cancellation message prints on the printer. If /T, **All files canceled by operator**. If /C, the name of the file followed by **File canceled by operator**, where File is the name of the file.
- The printer paper advances to the next page.
- The printer alarm sounds.
- If all files in the print queue have not been canceled, printing resumes with the first file remaining in the print queue.

Notes:

1. The disk containing the files being printed must remain in the specified drive until all printing is complete. Any file in the print queue must not be altered or erased until after it has been printed.

2. The printer cannot be used for any other purpose while PRINT has data to print. Any attempt to use the printer (Shift-PrtSc, LLIST, LPRINT, etc.) results in an "out-of-paper" or "time-out" indication until all files have been printed or printing is terminated (/T). Using Ctrl-PrtSc will result in a "not ready" error message. You should press Ctrl-PrtSc again and reply A to the error message.

Example: In this example, the PRINT command is being used for the first time since system was started. The command:

```
PRINT a:temp1.tst
```

has just been entered, DOS responds with:

```
Name of list device [PRN]:
```

Press the Enter key to send output to the printer.

Then it adds the file TEMP1.TST from drive A to the print queue and sends the output to the device "PRN" printer. The command:

```
PRINT /T
```

empties the print queue. Any other information on the line is ignored. The command:

```
PRINT temp.*/C
```

removes all TEMP.??? files from the print queue that have the same drive letter as the default drive. The command:

```
PRINT a:temp1.tst/C a:temp2.tst a:temp3.tst
```

removes the three files TEMP1, 2, and 3 on drive A from the print queue. The command:

```
PRINT temp1.tst/C temp2.tst/P temp3.tst
```

removes file TEMP1.TST from the print queue, adds the files TEMP2.TST and TEMP3.TST to the print queue. The command:

```
PRINT temp1.tst temp2.tst temp3.tst/C
```

adds files TEMP1.TST and TEMP2.TST to the print queue, then removes TEMP3.TST from the print queue.

PROMPT (Set System Prompt) Command

Purpose:	Sets a new system prompt.
Format:	PROMPT [*prompt-text*]

Type: Internal External ***

Remarks: All text on the PROMPT command line is taken by DOS to be the new system prompt. If no parameter is specified, the normal DOS prompt is assumed. Special meta-strings can be imbedded in the text in the form $*c*.

Where *c* is one of the following:

$	The "$" character.
t	The time.
d	The date.
p	The current directory of the default drive.
v	The version number.
n	The default drive letter
g	The ">" character.
l	The "<" character.
b	The " ¦" character.
q	The "=" character.
h	A backspace and erasure of the previous character.
e	The ESCape character.
–	The CR LF sequence (go to beginning of new line on the display screen).

Any other character is treated as a null character—no action is taken on it by the PROMPT command.

Example: In this example, the command would set the normal DOS prompt:

```
PROMPT $n$g
```

In this example, the command would set ABC as the system prompt:

```
PROMPT ABC
```

In this example, the command would set a two line prompt that displays:

```
Time = (current time)

Date = (current date)

PROMPT Time = $t$_ Date = $d
```

If you wish to create a prompt that begins with any of the DOS command delimiters (such as semicolon, blank, etc.), you can precede that character with a null meta-string. The character will then be treated as the first character of the prompt, rather than as a delimiter between the word PROMPT and its parameter. For example:

```
PROMPT $A;ABC
```

causes the PROMPT command to interpret the $A as a null character because A is not one of the defined characters in the above list. All characters following the null character will become the new system prompt.

RECOVER Command

Purpose:	Recovers files from a disk that has developed a defective sector. You can recover the file that contains the bad sector (minus the data in the bad sector), or all the files on the disk can be recovered if the directory has been damaged.
Format:	RECOVER [*d*:][*path*]*filename*[.*ext*] or RECOVER *d*:
Type:	Internal External ***
Remarks:	The file named by *filename* is the file to be recovered. If you do not specify a drive, the default drive is used. If you do not specify a path, the current directory is used. The size of the recovered file is a multiple of the DOS allocation unit size. In most cases, this is larger than the original file size. Text files will normally require re-editing to remove unwanted data from the end of the recovered file before they can be used for normal processing.

In the second format shown, RECOVER assumes the directory is damaged, and recovers all files on the specified disk.

If the global filename characters * and ? are used in the filename or extension, only the first file that matches the filespec will be recovered. RECOVER only recovers one file at a time when a filespec is entered.

Example: For this example assume the disk file to be recovered is MYPROG:

```
RECOVER A:MYPROG
```

This command causes the disk file MYPROG on drive A to be read sector-by-sector, skipping the bad sectors. The bad sectors are allocated in a system table, thus preventing future allocations of that sector. The filename is not changed.

The following example shows how to recover the contents of an entire disk from drive A:

```
RECOVER A:
```

This command causes the disk file allocation table on drive A to be scanned for chains of allocation units. A new root directory is created for each chain of allocation units in the form:

FILE*nnnn*.REC

Where *nnnn* is a sequential number starting with 0001. Each FILE*nnnn*.REC points to one of the recovered files on the disk.

> Note: This form of the RECOVER command should only be used if the directory of the disk becomes unusable. Because RECOVER has no way of knowing whether the data in the directory is valid or not, it *must* assume that the entire directory is invalid, and therefore recovers all files into filenames of the forms shown above, including any files for which there may still have been valid directory entries.

RENAME (or REN) Command

Purpose:	Changes the name of the file specified in the first parameter to the name and extension given in the second parameter. If a valid drive is specified in the second parameter, the drive is ignored.
Format:	REN[AME] [*d*:][*path*]*filename*[.*ext*] *filename*[.*ext*]
Type:	Internal *** External
Remarks:	You can use the abbreviated form REN for the RENAME command. You can also use the global characters ? and * in the parameters. For more information about global characters, refer to "Global Filename Characters" in Chapter 1. A path can be specified only with the first filename; the file will remain in the same directory after its name has been changed.
Example:	The command: **RENAME B:ABODE HOME** renames the file ABODE on drive B to HOME.

The command:

```
REN B:ABODE *.XY
```

renames the file ABODE on drive B to ABODE.XY.

The command:

```
REN B:\LEVEL2\MYPROG.COM MYPROG1.COM
```

renames the file MYPROG.COM in directory \LEVEL2 on drive B to filename MYPROG1.COM.

RESTORE (Fixed Disk) Command

Purpose:	Restores one or more files from diskettes to a fixed disk.
Format:	RESTORE *d*: [*d*:][*path*][*filename*[.*ext*]][/S][/P]
Type:	Internal External ***

Remarks: The files being restored must have been placed on the diskettes by the BACKUP command. The first parameter you specify is the backup diskette drive. The second parameter is the fixed disk file you want to restore.

Files are restored to the current directory if you do not specify a path. If you do not specify a filename or extension, then all files backed up from the directory will be restored.

Global filename characters are allowed in the filename, and cause all of the files matching the filename to be restored. For example, entering:

```
RESTORE A: C:*.DAT
```

restores each file from the backup diskettes with an extension of .DAT that had been backed up from the current directory.

The parameter **/S** causes backed up files in all subdirectories to be restored in addition to the files in the specified directory itself. This includes subdirectories at all levels beyond the specified directory.

The parameter **/P** causes RESTORE to prompt you before restoring files that have changed since they were last backed up, or that are marked read-only. You can then choose to restore the file or not. Read-only is a file attribute that an application can set by interfacing with DOS internally. The two DOS system files (IBMBIO.COM and IBMDOS.COM) are marked read-only when they are created by the FORMAT and SYS commands.

The following example restores all files on the backup diskettes to fixed disk drive C:

```
RESTORE A: C:\ /S
```

The next example restores three different files from the backup diskettes to the default fixed disk drive:

```
RESTORE A: \level1\file1.dat
RESTORE A: \level1\level2\file2.dat
RESTORE A: \level1\level3\file3.dat
```

When RESTORE prompts you to insert the backup diskette, make sure you insert the first diskette that might contain the file you want to restore. If you are not sure, insert backup diskette number one. If the file is not on the diskette you inserted, RESTORE will prompt you to insert the next diskette.

If you used global filename characters, RESTORE prompts you to insert the next diskette after it has restored all files on the backup diskette that match the specified filename.

The RESTORE command sets the ERRORLEVEL (see Batch Commands) as follows:

0	Normal completion
1	No files were found to restore
3	Terminated by user (Ctrl-Break or ESC)
4	Terminated due to error

These codes can be used with the batch processing IF subcommand to control subsequent error level processing.

RMDIR (Remove Directory) Command

Purpose: Removes a subdirectory from the specified disk.

Format: RMDIR [*d*:]*path*

or

RD [*d*:]*path*

Type:

Internal	External
***	

Remarks: The directory must be empty before it can be removed with the exception of the "." and ".." entries. The last directory name in the path is the directory to be removed.

Example: In this example, the command:

```
RD B:\LEVEL2\LEVEL3
```

removes the entry for LEVEL3 from directory LEVEL2.

> **Note:** The root directory and the current directory cannot be removed.

SET (Set Environment) Command

Purpose: This command inserts strings into the command processor's environment. The entire series of strings in the environment is made available to all commands and applications.

Format: SET [*name*=[*parameter*]]

Type:

Internal	External
***	

Remarks: The entire string (beginning with *name*) is inserted into a block of memory reserved for environment strings. Any lowercase letters in the name are converted to uppercase letters when added to the environment; the remainder of the line is inserted as you entered it. If the name already existed in the environment, it is replaced with the new *parameter*.

If the SET command is entered with no *name* specified, then the current set of environment strings will be displayed.

If a *name* is specified, but the *parameter* is not specified, then the current occurrence of *name*=*parameter* is removed from the environment.

SET (Set Environment) Command

The environment (series of names and parameters) is made available to all DOS commands and applications (see the "Program Segment Prefix" description in Chapter 6 of the IBM *DOS Technical Reference*). You can display the current environment contents by entering a SET command with no parameters. You can select the strings in the environment. For example, entering:

```
SET abc=xyz
```

will add the string ABC=xyz to the other strings already in the environment (note the conversion of abc to uppercase ABC). In this way, it is possible for you to enter keywords and parameters that are not meaningful to DOS, but can be found and interpreted by applications that are designed to examine the environment.

Example: This example adds the string PGMS=\LEVEL2 to the environment. When an application program receives control, it could search the environment for the name PGMS, and use the supplied parameter as the directory name to use for its files:

```
SET PGMS=\LEVEL2
```

The following example would remove PGMS=\LEVEL2 from the environment:

```
SET PGMS=
```

Notes:

1. DOS automatically adds any PROMPT or PATH commands to the environment when you enter them. You do not need to use the SET command to add either of these two commands to the environment.

2. One of the strings in the environment (placed there by DOS when it starts up) will always be a COMSPEC = parameter. That parameter describes the path that DOS uses to reload the command processor when necessary.

3. If you have *not* loaded a program that remains resident (such as MODE, PRINT, GRAPHICS, etc.), DOS expands the environment string area to hold additional strings. If you *have* loaded a program that remains resident, DOS is unable to expand the environment area beyond 127 bytes or if the environment area has already expanded beyond 127 bytes when you load a program that is to remain resident, DOS is unable to expand the environment area beyond that point. The message **Out of environment space** appears if you issue a SET command that would cause the combined environment strings to exceed 127 bytes.

SORT Filter Command

Purpose:	This filter command reads data from the standard input device, sorts the data, then writes the data to the standard output device.
Format:	SORT [/R] [/+*n*]

Type:	Internal	External
		***

Remarks: Sorts are done using the ASCII collating sequence. Tab characters are not expanded with blanks.

The /R parameter will reverse the sort, for example make "Z" come before "A."

The /+*n* parameter is an integer that starts the sort with column *n*. If no parameters are specified, the sort starts with column 1. The maximum file size that can be sorted is 63K.

Example: In this example, the command line will read the file UNSORT.TXT, do a reverse sort, then write the output to file SORT.TXT:

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

In the next example, the command line causes the output of the directory command to be piped to the SORT filter. The SORT filter will sort starting with column 14 (this is the column the file size starts in), then send the output to the console. Thus, a directory sorted by file size will be the result:

```
A>DIR | SORT /+14
```

SYS (System) Command

Purpose: Transfers the operating system files from the default drive to the specified drive.

Format: SYS *d*:

Type:	Internal	External
		***

Remarks: The directory of the disk in the specified drive must be completely empty, or the disk must have been formatted by a FORMAT *d*:/S or FORMAT *d*:/B command to contain directory entries for the DOS files IBMBIO.COM and IBMDOS.COM. This is necessary because DOS startup requires these files to occupy the first two directory entries, and because IBMBIO.COM must reside on consecutive sectors on the disk.

Note: SYS lets you transfer a copy of DOS to an application program diskette designed to use DOS, but sold without it. In this case, the space required for the DOS files has already been allocated, although the DOS files are not actually present. The SYS command will transfer the files to the allocated space.

TIME Command

Purpose: Permits you to enter or change the time known to the system. Whenever you create or add to a file, the time is recorded in the directory. You can change the time from the console or from a batch file.

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

Type:	Internal	External
	***	

Remarks: If you enter a valid time with the TIME command, the time is accepted, and the system prompt appears. Otherwise, the TIME command displays the following prompt:

```
Current time is hh:mm:ss.xx
Enter new time:_
```

where:

hh	is a one- or two-digit number from 0-23 (representing hours)
mm	is a one- or two-digit number from 0-59 (representing minutes)
ss	is a one- or two-digit number from 0-59 (representing seconds)
xx	is a one- or two-digit number from 0-99 (representing hundredths of a second)

Notes:

1. To leave the time as is, press Enter.

2. If you enter any information (for example, just the hours, and press Enter), the remaining fields are set to zero.

3. Any time is acceptable as long as the digits are within the defined ranges.

4. The valid delimiters within the time are the colon (:) separating the hours, minutes, and seconds, and the period (.) separating the seconds and the hundredths of a second.

5. If you specify an invalid time or delimiter, you receive an **Invalid time** message.

Example: In this example, once you press Enter, the time known to the system is changed to 13:55:00.00.

```
A>TIME
Current time is 00:25:16.65
Enter new time: 13:55_
```

TREE (Display Directory) Command

Purpose: Displays all of the directory paths found on the specified drive, and optionally lists the files in each sub-directory.

Format: TREE [*d*:][/F]

Type: Internal External
***

Remarks: If no drive is specified, the default drive is assumed.

For each directory found, its full path name will be displayed, along with the names of any directories defined within it (these are called subdirectories in the output). If the /F parameter is used, the names of all files in each subdirectory are also displayed.

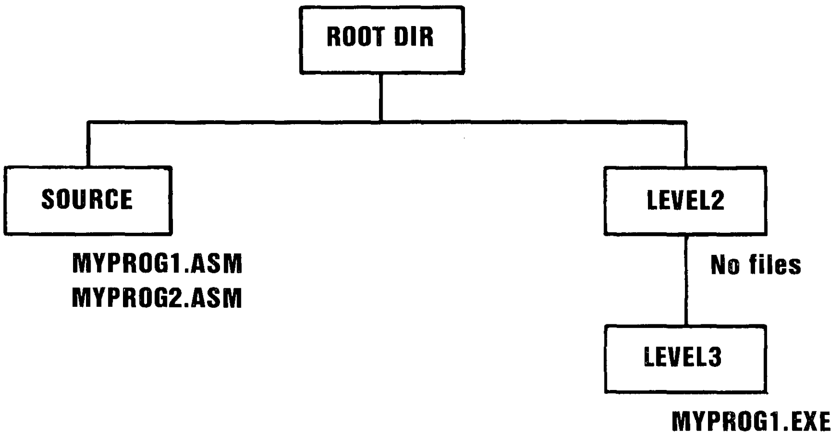
To make the screen output pause, use the Ctrl-Num Lock keys or pipe the output to the MORE filter.

Example: In this example, the command:

```
TREE B:/F >TREE.LST
```

causes all directories on drive B to be listed. The output will be placed in file TREE.LST in the current directory of drive B, and will contain the names of all subdirectories and files at each directory level.

Following is an example of a directory path listing. If the disk called MYDISK in drive A had the following directory structure:



Then TREE would display:

```
DIRECTORY PATH LISTING FOR VOLUME MYDISK

Path: \SOURCE

Subdirectories: None

Files:      MYPROG1 .ASM
            MYPROG2 .ASM


Path: \LEVEL2

Subdirectories: LEVEL3

Files:      None


Path: \LEVEL2\LEVEL3

Subdirectories: None

Files:      MYPROG1 .EXE
```

The following example lists all the subdirectories and filenames from drive A on the printer:

```
TREE A:/F >PRN
```

TYPE Command

Purpose:	Displays the contents of the specified file on the standard output device.
Format:	TYPE [*d*:][*path*]*filename*[.*ext*]

Type: Internal External
***

Remarks: The data is unformatted except that tab characters are expanded to an eight-character boundary; that is, columns 8, 16, 24, etc.

Notes:

1. Press Ctrl-PrtSc if you want the contents of a file to be printed as it is being displayed. You can also redirect the output to a file or the printer.

2. Text files appear in a legible format; however, other files, such as object program files, may appear unreadable due to the presence of nonalphabetic or nonnumeric characters.

3. You must specify a filespec.

4. Global filename characters are not allowed in the filename or extension. If global filename characters are used in the filename or extension, the message **File not found** will appear.

Example: In this example, the file MYPROG.ONE on the diskette in drive B is displayed on the screen:

```
TYPE  B:myprog.one
```

VER (Version) Command

Purpose: Displays the DOS version number that you are working with on the display screen.

Format: VER

Type:	Internal	External
	***	

Remarks: The DOS version consists of a single-digit major version number, followed by a period, followed by a two-digit minor revision level.

Example:

```
A>VER
IBM Personal Computer DOS Version 2.10
```

VERIFY Command

Purpose: Verifies that the data written on a disk has been correctly recorded.

Format: VERIFY [ON | OFF]

Type:

Internal	External
***	

Remarks: VERIFY ON remains on until it is turned off through the SET VERIFY System Call or a VERIFY OFF command.

When ON, DOS performs a verify operation following each disk write operation, to verify that the data just written can be read without error. Because of the extra time required to perform the verification, the system runs slower when programs write data to disk.

Entering VERIFY with no parameters causes DOS to display the current state (on or off) of the verify feature.

Example: This example causes the verify feature to be turned on:

```
A>VERIFY ON
```

This example displays the current status:

```
A>VERIFY
VERIFY is on

A>
```

VOL (Volume) Command

Purpose: Displays the disk volume label of the specified drive.

Format: VOL [*d*:]

Type:	Internal	External
	***	

Remarks: If you do not specify a drive, the default drive is assumed.

Example:

```
A>VOL
Volume in drive A is MYDISK

A>
```

Summary of DOS Commands

The following chart is provided for quick reference. The section called "Format Notation" at the beginning of Chapter 1 explains the notation used in the format of the commands.

Note: In the column labeled Type, the I stands for Internal and the E stands for External.

Command	Type	Purpose	Format
(Batch)	I	Executes batch file	[*d*:]*filename* [*parameters*]
ECHO	I	Inhibits screen display	ECHO [ON | OFF| *message*]
FOR	I	Iterative execution of commands	FOR %%*variable* IN (*set*) DO *command*
GOTO	I	Transfers control to line following the label	GOTO *label*
IF	I	Conditional execution of commands	IF [NOT] *condition command*
SHIFT	I	Shift command lines	SHIFT
PAUSE	I	Provides a system wait	PAUSE [*remark*]

Figure 1 (Part 1 of 2). DOS Batch Processing Commands

Command	Type	Purpose	Format
REM	I	Displays remarks	REM [*remark*]

Figure 1 (Part 2 of 2). DOS Batch Processing Commands

Command	Type	Purpose	Format
ASSIGN	E	Routes requests to a different drive	ASSIGN [*x*=*y* [...]]
BACKUP	E	Backs up fixed disk files	BACKUP *d*:[*path*] [*filename*[.*ext*]] *d*: */S][/M][/A] [D:mm-dd-yy*]
BREAK	I	Checks for control break	BREAK [ON | OFF]
CHDIR	I	Change current directory	CHDIR [[*d*:]*path*] or CD [[*d*:]*path*]
CHKDSK	E	Checks disk and reports status	CHKDSK [*d*:][*filename*[.*ext*]] [/F][/V]
CLS	I	Clears the display screen	CLS
COMP	E	Compares files	COMP [*d*:][*path*][*filename* [.*ext*]] [*d*:][*path*][*filename* [.*ext*]]

Figure 2 (Part 1 of 6). DOS Commands

Command	Type	Purpose	Format
COPY	I	Copies files	COPY [/A][/B] [*d*:][*path*]*filename* [.*ext*][/A][/B] [*d*:][*path*][*filename* [.*ext*]][/A][/B][/V] or COPY [/A][/B] [*d*:][*path*]*filename* [.*ext*][/A][/B] [+[*d*:][*path*] *filename*[.*ext*] [/A][/B]...] [*d*:][*path*][*filename* [.*ext*]][/A][/B][/V]
CTTY	I	Change standard input and standard output device.	CTTY *device-name*
DATE	I	Enter date	DATE [*mm-dd-yy*]
DIR	I	Lists filenames	DIR [*d*:][*path*] [*filename*][.*ext*] [/P][/W]
DISKCOMP	E	Compares diskettes	DISKCOMP [*d*:] [*d*:] [/1][/8]
DISKCOPY	E	Copies diskettes	DISKCOPY [*d*:] [*d*:] [/1]

Figure 2 (Part 2 of 6). DOS Commands

Command	Type	Purpose	Format
ERASE	I	Deletes files	ERASE [*d*:][*path*][*filename* [*.ext*]] or DEL [*d*:][*path*][*filename* [*.ext*]]
EXE2BIN	E	Converts .EXE files to .COM format	EXE2BIN [*d*:][*path*]*filename* [*.ext*]] [*d*:][*path*][*filename* [*.ext*]]
FIND	E	Searches files for strings of text	FIND [/V][/C][/N] *string* [[*d*:][*path*] *filename*[*.ext*]...]
FORMAT	E	Formats diskette	FORMAT [*d*:][/S] [/1][/8][/V][/B]
GRAPHICS	E	Prints graphics display screen	GRAPHICS
MKDIR	I	Creates a subdirectory	MKDIR [*d*:]*path* or MD [*d*:]*path*

Figure 2 (Part 3 of 6). DOS Commands

Command	Type	Purpose	Format
MODE	E	Sets mode on printer/display	MODE LPT#:[*n*] [,[*m*] [,P]] or MODE *n* or MODE [*n*],*m*[,T] or MODE COM*n*: *baud*[,*parity* [,*databits*[,*stopbits* [,P]]]] or MODE LPT#:= COM*n*
MORE	E	Displays a screen full of data	MORE
PATH	I	Searches directories for commands or batch files	PATH [[*d*:]*path*[[;[*d*:] *path*]...]]
PRINT	E	Queues and prints data files	PRINT [[*d*:][*filename*[.*ext*]] [/T][/C][/P]...]

Figure 2 (Part 4 of 6). DOS Commands

Command	Type	Purpose	Format
PROMPT	E	Set new prompt	PROMPT [*prompt-text*]
RECOVER	E	Recovers files from disk or diskette	RECOVER [*d*:][*path*]*filename* [.*ext*] or RECOVER *d*:
RENAME	I	Renames files	REN[AME] [*d*:][*path*]*filename* [.*ext*]*filename*[.*ext*]
RESTORE	E	Restores diskette files to fixed disk	RESTORE *d*: [*d*:][*path*][*filename* [.*ext*]][/S][/P]
RMDIR	I	Removes a subdirectory	RMDIR [*d*:]*path* or RD [*d*:]*path*
SET	I	Inserts strings into the command processor's environment	SET [*name*= [*parameter*]]
SORT	E	Sorts text data	SORT [/R] [/+*n*]
SYS	E	Transfers DOS	SYS *d*:
TIME	I	Enter time	TIME [*hh:mm:ss.xx*]

Figure 2 (Part 5 of 6). DOS Commands

Command	Type	Purpose	Format
TREE	E	Displays all directory paths	TREE [*d*:][/F]
TYPE	I	Displays file contents	TYPE [*d*:][*path*\\]*filename* [.*ext*]
VER	I	Displays version number	VER
VERIFY	I	Verifies data	VERIFY [ON | OFF]
VOL	I	Displays volume identification	VOL [*d*:]

Figure 2 (Part 6 of 6). DOS Commands

Chapter 3. Preparing Your Fixed Disk

Contents

Introduction

If your IBM Personal Computer has a fixed disk, there are several facts you need to know, and several steps to take before DOS is able to use it. If you try to use your fixed disk before you take the following steps, you will get the error message:

```
Invalid drive specification
```

The first section in this chapter, "Fixed Disk Drive Letters" describes how drive letters are used with the fixed disk. The remaining sections describe how to prepare your fixed disk for DOS.

> **Note:** If your fixed disk is in an expansion unit (not in the system unit), the expansion unit must be turned on *before* you turn on the system unit.

A fixed disk can be divided up into separate areas called partitions. There can be from one to four partitions on a fixed disk. These partitions can be different sizes. Each partition is set up through a Fixed Disk Setup Program (called FDISK) provided by the operating system that will use it.

You will use the DOS Fixed Disk Setup Program to set up the DOS partition. Fixed disk drives are referred to in the same way you refer to diskette drives, using the fixed disk drive specifier when you want to read from or write to the fixed disk.

If DOS is the only operating system that you intend to use with the fixed disk, you will want to assign all of the fixed disk space for use with DOS. Follow the steps in "Preparing Your Fixed Disk" to assign all of the fixed disk space to the DOS partition.

If you do intend to use part of the fixed disk with other operating systems, then you will need to divide up the available fixed disk space among them. Follow the steps in "Partitioning Your Fixed Disk" to assign a specific amount of disk space to DOS.

If you are not sure whether you will be using any other operating system with the fixed disk, you should go ahead and assign all of the fixed disk space to DOS as described in "Preparing Your Fixed Disk." If you decide later to use another operating system, you can use the BACKUP command to backup your files in your DOS partition, reassign the DOS partition, and then use the RESTORE command to restore your files from diskettes. After you have followed the instructions in "Preparing Your Fixed Disk" or "Partitioning Your Fixed Disk," you will need to follow the instructions in "Setting Up the DOS Partition" in order to make the partition you have created useable by DOS.

The menus and screens that make up the Fixed Disk Setup Program have been designed to make it easy for you to set up your fixed disk. When the Fixed Disk Setup Program asks you to enter something, it displays a default answer for you. If that is the answer you want, you need merely press the Enter key. If you want to enter something else, simply type in the entry you want and then press the Enter key.

Fixed Disk Drive Letters

You already know that if you have two diskette drives on your system, they are known to DOS as A and B. If you have only one diskette drive on your system, there are still two diskette drives (A and B) known to DOS. But in this case, they are simulated on the same physical drive. (If you are not yet familiar with this concept, please refer to "How to Use DOS With One Diskette Drive" in Chapter 1 of the *DOS User's Guide* before you continue.)

When DOS starts, it first assigns letters to all of the diskette drives it knows about, then assigns the following letters to your fixed disks. For example, if you have one or two diskette drives, and one fixed disk, the letters A and B apply to the diskettes, and your fixed disk is known to DOS as C (if you had a second fixed disk, it would become D).

Preparing Your Fixed Disk

If DOS is the only operating system that you intend on using with your fixed disk, follow the instructions in this section. All of the fixed disk space will be assigned for use with DOS.

If you intend to use part of the fixed disk with another operating system, then you should go to "Partitioning Your Fixed Disk" in this chapter.

In order to prepare the fixed disk for use with DOS, you must first use the FDISK command as follows:

1. With your DOS diskette in drive A and the DOS prompt (A>) on the screen, type:

   ```
   FDISK
   ```

   and press Enter. The following screen appears:

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C)Copyright IBM Corp. 1983

FDISK Options

Current Fixed Disk Drive: 1

Choose one of the following:

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

Enter choice: [1]
```

The current fixed disk drive and option 5 will only be shown if your system has more than one fixed disk drive. If you want to set up the DOS partition on the next drive, type:

```
5
```

and press Enter. The Fixed Disk Setup Program will allow you to create an active partition on a fixed disk other than the first fixed disk. However, DOS can only be started from the first fixed disk (refer to “Changing the Active Partition” in this chapter to change the partition status). You will see the drive number change on the screen after you press Enter.

2. Type:

```
1
```

and press Enter to set up the fixed disk for use with DOS. If the fixed disk has not already been set up for DOS or another operating system, then the following screen appears:

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C)Copyright IBM Corp. 1983

Create DOS Partition

Current Fixed Disk Drive: 1


Do you wish to use the entire fixed disk
for DOS (Y/N).....................? [Y]
```

If your fixed disk has already been set up, then you will see a different screen that shows how the fixed disk partitions have been assigned, and you will get a different prompt. If this happens, you should follow the steps in “Partitioning Your Fixed Disk” in this chapter.

3. You should press Enter since you want to use the entire fixed disk for DOS. The Fixed Disk Setup Program will then assign the fixed disk to DOS, and display the following message:

```
Insert DOS diskette in drive A:
Press any key when ready . . .
```

4. You must now restart DOS so that it will recognize your fixed disk and assign a drive letter to it. With your DOS diskette in drive A, press any key to restart DOS.

Your fixed disk has now been set up with a DOS partition. But before DOS can use it, DOS needs to create a directory and other information in the partition. To do this, follow the instructions in "Setting Up the DOS Partition."

Setting Up the DOS Partition

The DOS fixed disk partition must be formatted by the DOS FORMAT command before it can be used. You should only follow these instructions if the DOS partition has been created, but has not already been formatted and used to store data. This is because any data in the partition will be destroyed by the format operation.

1. Make sure your DOS diskette is in drive A and the DOS prompt (A>) is on the screen.

   If DOS is to be started from the fixed disk, enter:

   ```
   FORMAT d:/S/V
   ```

   If the partition is not to contain a copy of DOS (not to be automatically started), enter:

   ```
   FORMAT d:/V
   ```

   In either case, substitute the correct fixed disk drive letter for the *d* in the command (for example, if you have 1 or 2 diskette drives, you would enter C, as shown in the following screens). This prompt now appears:

```
Press any key to begin formatting drive C:
```

2. Press any key. The red light on your fixed disk drive will light up, and the message:

```
Formatting...
```

appears on the screen. Do not be alarmed if several minutes go by before you see any more messages. DOS is checking the data in every location in the DOS partition and it takes several minutes. You will see the message:

```
Format complete
```

and, if you used /S in your FORMAT command, you also see the message:

```
System transferred
```

This tells you that a copy of DOS has been placed on the fixed disk.

Then the following message appears:

```
Volume label (11 characters, ENTER for none)?
```

3. Enter a 1-11 character *volume label* (for example, MYFIXEDDISK) that is used to identify the fixed disk when DIR and CHKDSK displays information. If you do not want to label the fixed disk, just press Enter; however, please note that you cannot add a volume label later, so we recommend entering one now.

   FORMAT then displays the disk space statistics and the DOS prompt:

```
A>
```

Your fixed disk is now completely usable by DOS.

If you have placed a copy of DOS on the fixed disk, we recommend that you do the following two steps:

4. With your DOS diskette still in drive A, enter:

   ```
   COPY *.* d:
   ```

   Remember to use a correct fixed disk letter for your system. This copies all the programs on the DOS diskette to your fixed disk. Once these programs are copied, all DOS commands can be run from the fixed disk and your DOS diskette can be stored away in a safe place.

   Also, remember to copy the programs from your DOS Supplemental Program diskette in the same manner, if you will be using these programs.

5. Remove the DOS diskette from drive A (leave the diskette drive door open) and press the Ctrl, Alt, and Del keys simultaneously (System Reset). If you have correctly followed the steps above, and a copy of DOS was stored in the DOS partition, DOS will start from the fixed disk and you will be asked to enter the date and time.

   > Note: Drive A must be empty (no diskette or the diskette drive door open) in order for this to work correctly. This is because the computer will first try to load an operating system from drive A. If a diskette cannot be read from drive A, then (and only then), the computer will try to load an operating system from the first fixed disk on the computer.

When you have entered the date and time, you will notice that the DOS prompt (A>) has now changed. Instead of the letter A>, you will see the drive letter of your fixed disk. DOS remembers which drive it was started from, and makes that drive the default drive.

If you have followed the instructions above, your fixed disk is now completely initialized.

Partitioning Your Fixed Disk

In order that more than one operating system can use the fixed disk, the fixed disk must be divided into separate areas called partitions. There can be from one to four partitions on the fixed disk. These partitions can be different sizes and can be set up in any order. You can specify which partition the system will get control of when you start or restart your computer. An operating system can only access one partition. You cannot transfer data directly from one partition to another.

Each operating system that supports the fixed disk provides a program to allow you to create a partition for use under that system. If you try to read from, or write to, the fixed disk using a system that has no partition assigned to it, you will get an error message.

The DOS Fixed Disk Setup Program can only be used to create or delete the DOS partition. A partition set up for another operating system can only be created or deleted from that operating system.

You can set up one partition for use under DOS at the location and size you choose. You can also delete the DOS partition if, for example, you want to create it again at a different size or location on the fixed disk. The following functions are supported by the DOS Fixed Disk Setup Program:

- Create the DOS partition.
- Change the active partition (the one that will be started when the system is restarted).

- Delete the DOS partition.
- Display fixed disk partition data.
- Select next fixed disk drive.

These functions are described below in separate sections. In order to get access to them, you need to start the Fixed Disk Setup Program as follows:

1. To start, type:

   *d:***FDISK**

   and then press Enter. Where *d*: is the drive where the FDISK.COM program resides.

2. If it is on the default drive, just type:

   FDISK

   and press Enter. You will then see the following screen:

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C)Copyright IBM Corp. 1983

FDISK Options

Current Fixed Disk Drive: 1


Choose one of the following:

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

Enter choice: [1]
```

The current fixed disk drive and option number 5 will be shown only if your system has more than one fixed disk drive. Option 5 selects the next fixed disk drive. The Fixed Disk Setup Program will allow you to create an active partition on a fixed disk other than the first fixed disk. However, DOS can only be started from the first fixed disk. The partition's status is shown as A for active and N for not active (refer to "Changing the Active Partition" in this chapter to change the partition status).

Type the number of the option you want and press the Enter key. Note that option 1 is the default and will automatically be selected if you don't type an option number before pressing Enter. Proceed to the section below that describes the option you selected.

Creating the DOS Partition (Option 1)

You can use this option to create the DOS partition. In order to do so, you need to determine where it should be located and how large it should be. If there is already a partition assigned to DOS, you will see an error message.

A fixed disk is divided into parts called cylinders. The number of cylinders and their sizes can vary depending on the fixed disk. A 10-megabyte disk contains 305 cylinders and each cylinder contains 34,816 bytes or characters of information. If you wanted to assign the whole fixed disk for use under DOS, you would specify the size as 305 cylinders and the starting cylinder number as 000.

The screen you see depends on whether the fixed disk has any partitions. If it has been initialized, go to step 2. If not, the following screen appears:

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C)Copyright IBM Corp. 1983

Create DOS Partition

Current Fixed Disk Drive:  1


Do you wish to use the entire fixed
disk for DOS (Y/N)..............................? [Y]
```

1. If DOS is the only operating system that you intend on using with the fixed disk, then you should follow the instructions in "Preparing Your Fixed Disk" in this chapter. Otherwise, type:

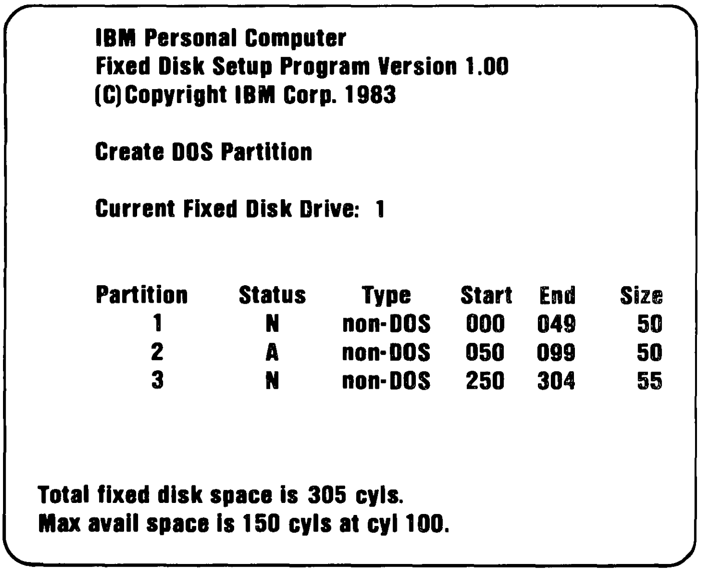
   n

   and then press the Enter key. The following message appears:

```
Total fixed disk space is xxxx cylinders.
Maximum available space is xxxx.
cylinders at cylinder  xxxx.
```

   These lines show the total number of cylinders on your fixed disk. Proceed to step 3.

2. You will see a screen similar to the following if your fixed disk has already been set up:



```
        IBM Personal Computer
        Fixed Disk Setup Program Version 1.00
        (C)Copyright IBM Corp. 1983

        Create DOS Partition

        Current Fixed Disk Drive: 1


        Partition    Status    Type     Start   End    Size
            1          N      non-DOS   000     049     50
            2          A      non-DOS   050     099     50
            3          N      non-DOS   250     304     55



    Total fixed disk space is 305 cyls.
    Max avail space is 150 cyls at cyl 100.
```

The "Create DOS Partition" screen shows a sample fixed disk with 3 partitions. Note that this is not necessarily a recommended setup. It is shown only as an example.

The line with the current fixed disk drive will only appear if you have more than one fixed disk drive.

There will be one line shown for each assigned partition.

The Partition column shows the relative number of the partition (in the order it appears on the fixed disk).

The status column shows which partition's system gets control when the system unit is started from the fixed disk. That partition's status is shown as A (for active), the others are shown as N (for not active).

The Type column shows which partition, if any, is the DOS partition.

The Start and End columns show the starting and ending cylinder numbers for a partition and the Size column shows its size in cylinders.

The next line shows you the total amount of space on the fixed disk, and the line after that shows you the size of the largest available space that you could use for a partition and where it is located on the fixed disk.

3. The following prompt appears:

```
Enter partition size............: [xxxx]
```

The partition size entry defaults to the largest available space on the fixed disk. If you want your DOS partition to use the largest available space, simply press the Enter key. Otherwise, type in the size you want (in cylinders) and press the Enter key. The next prompt is:

```
Enter starting cylinder number..: [xxxx]
```

4. The starting cylinder number default depends on the partition size you specified above. It is the first cylinder of the largest space on the fixed disk large enough for the partition. If you want the DOS partition to be located there, press the Enter key. Otherwise, type in the starting cylinder number you prefer and press the Enter key. The cursor is placed at the bottom of the screen and you see this message:

```
Press Esc to return to FDISK Options [ ]
```

Note that the lines on the screen change to show the new active partition. The DOS partition has now been created. With your DOS diskette in drive A, press the Ctrl, Alt, and Del keys simultaneously (System Reset).

If you need the partition you just created to be *active* (startable), follow the steps in "Changing Active Partition (Option 2)."

Your DOS partition has been created but you still need to follow the instructions in "Setting Up the DOS Partition" in this chapter before you can use the DOS partition.

Changing the Active Partition (Option 2)

Select this option when you want to start a different operating system in another partition. You will see a screen similar to the following:

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C)Copyright IBM Corp. 1983

Change Active Partition

Current Fixed Disk Drive:  1

Press Esc to return to Utility Options

Partition     Status     Type     Start   End     Size
      1          N      non-DOS    000    049       50
      2          N      non-DOS    050    149      100
      3          A        DOS      150    304      155

Total disk space is  xxxx  cylinders

Enter the number of the partition you
want to make active...............: [ ]
```

1. Enter the number of the partition whose operating system you want to get control when the system is started from the fixed disk. The following message appears:

```
Press Esc to return to FDISK Options [ ]
```

   Note that the lines on the screen change to show the new active partition.

2. Press the Esc key to return to the FDISK options menu and press it again to return to DOS.

   If you want to start the operating system in the partition you just made active, perform the following steps:

   a. Open the diskette drive A door.

   b. Press and hold Ctrl and Alt, and then press Del.

   The operating system in the active partition should then start.

Deleting the DOS Partition (Option 3)

Note: This option could result in loss of the data in the DOS partition so make sure you have backed up all of your files before you proceed.

1. You will need to insert a DOS diskette and restart the system from diskette drive A if you want to continue processing under DOS.

   If you want to start a system in another fixed disk partition, you should change the active partition to that partition number before you delete the DOS partition.

You will see a screen similar to the following:

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C)Copyright IBM Corp. 1983

Delete DOS Partition

Current Fixed Disk Drive:  1


Partition      Status     Type     Start   End     Size
    1            N       non-DOS    000    049      50
    2            N       non-DOS    050    099      50
    3            N         DOS      100    249     150
    4            A       non-DOS    250    304      55

Total fixed disk space is xxxx cylinders


Warning!  Data in the DOS partition
could be lost. Do you wish to
continue..................................? [N]
```

2. If you have backed up all of your files and are ready to continue, type Y and press Enter. If you decide to cancel the operation, press either the Enter key or the Esc key to return to the FDISK options menu.

   If you type Y and press Enter, the partition information displayed on the screen is updated, and the following message appears:

```
Press Esc to return to FDISK Options [ ]
```

The DOS partition has now been deleted. You will need to start another system from the fixed disk or restart DOS from a diskette to proceed.

Displaying Partition Data (Option 4)

You can use this option to display fixed disk status information. Your screen appears similar to the following:

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C)Copyright IBM Corp. 1983

Display Partition Information

Current Fixed Disk Drive: 1


  Partition  Status    Type     Start  End  Size
      1         A       DOS      000   199  200
      2         N     non-DOS    200   304  105



Total fixed disk space is xxxx cylinders
```

The line with the current fixed disk drive appears only if you have more than one fixed disk drive.

One line is shown for each assigned partition.

The Partition column shows the relative number of the partition (in the order it appears on the fixed disk).

The Status column shows which partition's system gets control when the system unit is started from the fixed disk. That partition's status is shown as A (for active), the others are shown as N (non-active).

The Type column shows which partition, if any, is the DOS partition.

The Start and End columns show the starting and ending cylinder numbers for a partition, and the Size column shows its size in cylinders.

The next line shows you the total amount of space on the fixed disk.

Press the Esc key when you are ready to return to the FDISK options menu.

Selecting Next Fixed Disk Drive (Option 5)

Select this option when you want to use the DOS Fixed Disk Setup Program with the next fixed disk drive.

After you have entered the option, you see the current fixed disk drive number change on the FDISK options menu.

This option is available only if your system has more than one fixed disk drive.

Chapter 4. Configuring Your System

Contents

Introduction

Each time DOS is started, it searches the root directory of the drive (from which it was started) for a special configuration file named CONFIG.SYS. If found, it reads the file and interprets the text commands within it.

Configuration Commands

The following commands can be included in the configuration file. If you add or change any of the configuration file commands, they will become effective the *next* time DOS is started.

BREAK Command

```
BREAK=ON/OFF
```

This command should only be used once in the configuration file. The default value is OFF, and causes DOS to check for Ctrl-Break being entered at the standard input device only when DOS is performing an operation involving:

- A standard output device
- A standard input device
- A standard print device

  or

- An Asynchronous Communication Adapter

With this setting, it may not be possible to cancel an executing program by using Ctrl-Break unless the program causes DOS to perform one of those four operations. Specifying ON causes DOS to check for Ctrl-Break whenever it performs *any* function for a program. This allows you to "break" out of programs that perform few (or no) standard output device, standard input device, standard print device, or auxiliary device operations (such as compilers). The ON/OFF state set in the configuration file can later be changed by issuing a BREAK command (see Chapter 2).

BUFFERS Command

BUFFERS=*xx*

Where *xx* is a number between 1 and 99. This is the number of disk buffers that DOS should allocate in memory when it starts up. The default value is 2, and this value will remain in effect until DOS is restarted with a different value specified in the configuration file.

What Is a Buffer?

A disk buffer is a block of memory that DOS uses to hold data being read from, or written to a disk (fixed disk or diskette), when the amount of data being transferred is not an exact multiple of the sector size. For example, if an application reads a 128-byte record from a file, DOS will read the entire sector into one of its buffers, locate the correct 128-byte record in the buffer, and move the record from the buffer into the application's area of memory. It then marks that buffer as having been used recently. On the next request to transfer data, DOS will attempt to use a different buffer. In this way, all of the buffers will eventually contain the most recently-used data. The more buffers DOS has, the more data will be in memory.

Read/Write Requests

Each time DOS is requested to read or write a record that is not an exact multiple of the sector size, it first looks to see if the sector containing that record is already in a buffer. If not, it must read the sector as described above. But if the data is already in a buffer, then DOS can simply transfer the record to the application's area without the need to read the sector from the disk—this saves time. This savings is realized on both reading and writing records, since DOS must first read a sector before it can insert a record your application is attempting to write.

Random/Sequential Applications

For applications that read and write records in a random fashion (such as many Basic and data base applications), the likelihood of finding the correct record already in a buffer increases if DOS has more buffers to work with. This can greatly speed up the performance of those applications.

For applications doing sequential reads and writes, however (read an entire file, write an entire file), there is little advantage to having a large number of buffers allocated.

Because all applications are different, there is no specific number of buffers that will serve all applications equally well. If your applications do little random reading and writing of records, the system default of 2 buffers (if you do not specify BUFFERS= in your configuration file) should be sufficient.

However, if you use data-base type applications, or run programs that perform a lot of random reads and writes of records, you will want to increase the number of DOS buffers. The "best" number of buffers for your particular application can only be determined by using different values until the best performance is achieved. For most data base applications, a value between 10 and 20 buffers will usually provide the best results.

Beyond that point, the system may appear to start running slower—this is because, with a very large number of buffers it can take DOS longer to search all the buffers for the record than it would take to read the record from disk.

Size of Your Computer

The final consideration in determining the number of buffers to allocate is the memory size of your computer. Since each additional buffer increases the resident size of DOS by 528 bytes, the amount of memory available to the application is reduced by that amount. Therefore, additional buffers may actually cause some applications to slow down, since there is less memory in which the application itself can keep data—this could result in more frequent reads and writes than would otherwise be necessary.

In summary, the optimum number of buffers must be determined by *you*, based on:

1. The types of applications most often used
2. The memory size of your computer
3. Your analysis of system performance when using your applications with different numbers of buffers allocated.
4. For computers with fixed disks, we recommend a minimum of BUFFERS=3.

DEVICE Command

DEVICE=[*d*:] [*path*]*filename*[.*ext*]

This command allows you to specify the name of a file containing a device driver. During startup, DOS loads the file into memory as an extension of itself, and gives it control as described in "Installation of Device Drivers" in Chapter 3 of *DOS Technical Reference*. Please refer to that section for technical information about installable device drivers.

Loading Standard Device Drivers

The standard device drivers loaded by DOS support the standard screen, keyboard, printer, auxiliary device, diskette, and fixed disk devices. A clock driver is also loaded (see Chapter 3 of the *DOS Technical Reference* manual). You don't need to specify any DEVICE= commands for DOS to support these devices.

Replacing Standard Device Drivers

If you wish to use the "Extended Screen and Keyboard Control" features described in Chapter 2 of the *DOS Technical Reference* manual, you should create the file CONFIG.SYS on the disk you will be starting DOS from. The file should contain the command DEVICE=ANSI.SYS. This command causes DOS to replace the standard screen and keyboard support with the extended functions.

Installing Your Own Device Driver

For systems programmers and application developers—if you have written device drivers that you want DOS to load when it starts, include a DEVICE= command in the CONFIG.SYS file for each driver to be loaded.

FILES Command

FILES=*xx*

The maximum value for *xx* is 99.

Beginning with DOS Version 2.00 and 2.10, there is no need for an application to construct a special control block (FCB) in order to access a file. Instead, the program can simply specify an ASCII string consisting of drive specifier, complete directory path name and filename when opening or creating a file. DOS will locate the correct drive, directory and file, and will create and return a *handle*–merely a 16-bit binary value.

Accessing a File

All file accesses (reads, writes, close) can then be performed by telling DOS which handle to use. When an application opens a file in this manner, DOS constructs a control block in its own memory on behalf of the application, in an area that was set aside when DOS started. The size of this area (and consequently, the maximum number of files that can be concurrently open), depends on the value specified in the FILES= command.

The default value is FILES=8; that is, no more than 8 files can be open at the same time. There is no effect on the number of files that can be concurrently open using the traditional (OPEN FCB) functions. This default value is sufficient for the majority of operating environments. However, if applications are installed that result in error messages indicating an insufficient number of handles, the FILES= command should be used to provide DOS with additional handles.

Number of Files Opened

The value specified in FILES= becomes the new maximum number of files that DOS allows to be concurrently open.

Note that this value is the maximum number of files allowed for the entire system. The maximum number of files that a process can have concurrently open is 20 (this number includes the 5 predefined handles for standard input, output, error, auxiliary, and standard printer).

If you specify FILES= in your configuration file, the size of the resident portion of DOS increases by *39* bytes for each additional file above the default value of 8. Consequently, the memory available to the application is reduced by the same amount. See function calls hex 3C through hex 46 in Chapter 5 of the *DOS Technical Reference* manual for descriptions of the new file-handling functions.

SHELL Command

SHELL=[*d:*] [*path*]*filename*[*.ext*]

This command allows you to specify the name and location of a top-level command processor that DOS initialization will load in place of COMMAND.COM.

System programmers who develop their own top-level command processor should remember to include provisions for handling interrupts hex 22, hex 23 and hex 24, and for reading and executing commands. Because the internal commands, batch processor, and EXEC function call (program loader) reside in COMMAND.COM., these functions will not be available to the user unless they are duplicated in your command processor.

Chapter 5. Using Tree-Structured Directories

Contents

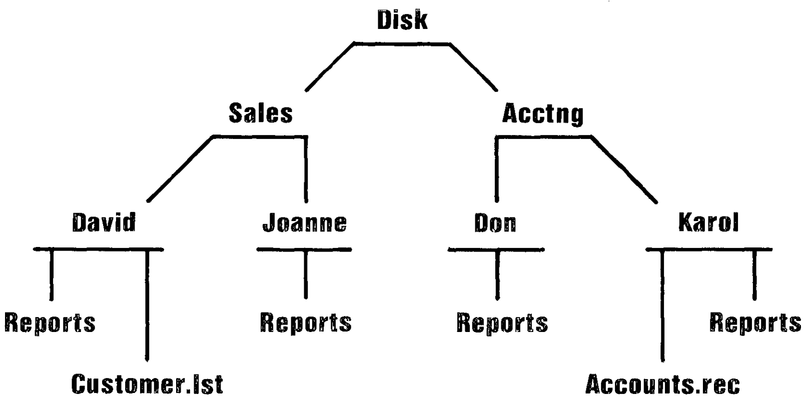
Introduction

Prior to Version 2.00 and 2.10, DOS used a simple directory structure that was adequate for managing files on diskettes. Each diskette contained a single directory that could hold a maximum of 64 or 112 files, depending on whether the diskette was single or dual sided.

With the added support for fixed disks in DOS Version 2.00 and 2.10, however, a single fixed disk can hold literally thousands of files. Keeping a large number of files in one directory becomes inefficient for both you and DOS (the larger a directory is, the longer it can take DOS to search for a file).

DOS Version 2.10 gives you the ability to better organize your disk by placing groups of related files in their own directories—all on the same disk (fixed disk or diskette).

For example, let's assume that the XYZ company has two departments (sales and accounting) that share an IBM Personal Computer. All of the company's files are kept on the computer's fixed disk. The logical organization of the file categories could be viewed like this:



With DOS Version 2.10, it is possible to create a directory structure that matches the file organization. With this ability, all of DAVID's report files can be grouped together in a single directory (called REPORTS), separated from all the other files on the disk. Likewise, all of the accounts receivable files can be in a unique directory, and so on.

Directory Types

As in previous versions of DOS, a single directory is created on each disk when you FORMAT it. That directory is called the root directory, or system directory.

A root directory on diskette can hold either 64 or 112 files—the maximum number of files in a fixed disk root directory depends on the size of the DOS partition on the disk.

In addition to containing the names of files, the root directory can also contain the names of other directories; and these in turn can contain the names of other files and directories; and so on.

Unlike the root directory, these other directories *called subdirectories* are actually files, and are therefore not restricted in size—they can contain any number of files and subdirectories, limited only by the amount of available space on the disk.

The subdirectory names are in the same format as filenames—a name of 1-to-8 characters optionally followed by a period and an extension of 1-to-3 characters. All characters that are valid for a filename are also valid for a directory name. Each directory can contain file and directory names that also appear in other directories. In other words, two or more files or directories can have the same name, as long as they are defined in separate directories.

The Current Directory

Just as DOS remembers a default drive, it can also remember a default directory for each drive on your system. This is called the *current directory,* and is the directory that DOS will search if you enter a filename without telling DOS which directory the file is in. You can change the current directory or find out what your current directory is for any drive by issuing the CHDIR command (described in Chapter 2). When DOS is started, it will automatically use the root directory as the current directory for each drive until you issue a CHDIR command.

Specifying the Path To a File

When you want DOS to create or search for a file, DOS must know three things—the drive, the name of the file, and the name of the directory containing the file.

If the file is in the current directory, you don't need to specify a directory—DOS automatically looks in the current directory.

But if the file is not in the current directory, you must supply DOS with the *path* of directory names leading to the desired directory. The path you specify can be either the path of names starting with the root directory, or the path from the current directory.

The *path* consists of a series of directory names separated by backslashes (\). If a filename is included, it must also be separated from the last directory name by a backslash.

If a path begins with a backslash, DOS starts its search from the root directory; otherwise, the search begins at the current directory.

For example, if the current directory is DAVID, and you want to find file ANNUAL.FIG in DAVID's REPORTS directory, you can specify it in either of these ways:

```
\SALES\DAVID\REPORTS\ANNUAL.FIG
```

or

```
REPORTS\ANNUAL.FIG
```

In the first case, the full path from the root directory (leading backslash) was specified. In the second case (no leading backslash), the path from the current directory was given.

Each subdirectory contains two special entries—you'll see them listed when you use the DIR command to list a subdirectory. The first contains a single period instead of a filename—it identifies this "file" as a subdirectory. The second entry contains two periods instead of a filename, and is used by DOS to locate the higher level directory that defines this directory, the *parent* of this directory. For example, in the file organization illustration shown on the first page of this chapter, the parent of the directory named JOANNE is the one named SALES.

This second special entry can be quite useful when you specify a path to DOS, because entering two periods is a shorthand way of telling DOS to *back up* one directory level. For example, if the current directory is DAVID, and you want to find file SUMMARY in JOANNE's REPORTS directory, you can specify it in either of these ways:

```
\SALES\JOANNE\REPORTS\SUMMARY
```

or

```
..\JOANNE\REPORTS\SUMMARY
```

The second case causes DOS to back up one level from the current directory (to the current directory's parent), and to continue the path from there. The double period can be used more than once in a path—it simply causes DOS to back up one level each time it is specified.

You can specify which drive to use by including a drive specifier ahead of the path and filename string. For example:

```
B:\LEVEL1\MYFILE
```

Notice that when defining a path, the drive is specified ahead of the path, rather than the filename.

Nearly all of the DOS commands that accept filenames also accept path names. For example, if you enter:

```
DIR  \ACCTNG\KAROL\REPORTS
```

all of the files in KAROL's REPORTS directory are listed. Similarly, if you enter:

```
DEL  \ACCTNG\KAROL\REPORTS
```

DOS assumes you want to erase all the files in KAROL's REPORT directory.

In all cases, to refer to a specific file, simply add the filename to the end of the path (separated from the path by a backslash, of course).

To refer to the root directory (if the root is not the current directory), enter one backslash. For example, if the current directory is DAVID, issuing:

```
DIR   \
```

will list all the files in the root directory.

You can create as many subdirectories as you wish. However, you should ensure that the longest path you create (from the root to the last directory in a single path) can be expressed in 63 characters or less.

Directory Commands

The following commands,. new in DOS Version 2.00 and 2.10, are included to help you create and manage your directory structure. A brief explanation of each is presented here–for more detailed information, please refer to the individual command descriptions in Chapter 2.

System programmers and application developers should also refer to the IBM *DOS Technical Reference*, particularly Chapter 5 for descriptions of the DOS function calls.

Creating a Subdirectory

The MKDIR (MD) command is used to create new directories. Be sure to include the appropriate drive and path, ending with the name of the new directory you want created. Also, be sure that the full path from the root to the new directory name is 63 characters or less.

Deleting a Directory

Directories can be deleted (removed) only with the RMDIR (RD) command. They cannot be deleted with the ERASE or DEL commands. A directory can be removed only if it is empty—that is, it has no files or subdirectories other than the two special entries shown as . and .. in the DIR command display. The last directory name in the specified path is removed—only one directory at a time can be deleted. The root directory and the current directory cannot be deleted.

Displaying and Changing the Current Directory

The CHDIR (CD) command is used to tell DOS which directory path it should "remember" as the current directory. Enter just a backslash for the root, or a full path for any other directory. The current directory is where DOS will look to find commands and files whose names are entered without path specifiers.

Entering CHDIR or CD with no parameters (or only a drive specification) causes DOS to display the current directory for the drive.

Displaying the Directory Structure

The TREE command will produce a report describing the entire directory structure of a disk. Included in the report are all directory paths and, optionally, the names of all files in each subdirectory.

✓

Where DOS Looks for Commands and Batch Files

When you enter a command, DOS searches the current directory for it (if the command is not built-in). The PATH command allows you to specify a series of additional paths that DOS can search if it does not find the command in the current directory. The PATH command is described in Chapter 2.

Chapter 6. The Line Editor (EDLIN)

Contents

Introduction

In this chapter, you will learn how to use the Line Editor (EDLIN) program.

You can use the Line Editor (EDLIN) to create, change, and display source files or text files. Source files are unassembled programs in source language format. Text files appear in a legible format.

EDLIN is a line text editor that you can use to:

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

The text of files created or edited by EDLIN is divided into lines of varying length, up to 253 characters per line.

Line numbers are generated and displayed by EDLIN during the editing process, but are not actually present in the saved file.

When you insert lines, all line numbers following the inserted text advance automatically by the number of lines inserted. When you delete lines, all line numbers following the deleted text decrease automatically by the number of lines deleted. Consequently, line numbers always go consecutively from 1 through the last line number.

> **Note:** EDLIN will erase the original backup copy (.BAK) of the file when you issue an **E** (end edit) command, or if the disk space is required during the editing session to satisfy a **W** (write lines) command.

How to Start the EDLIN Program

To start EDLIN, enter:

```
EDLIN [d:][path]filename[.ext][/B]
```

Editing an Existing File

If the specified file exists on the designated or default drive, the file is loaded into memory until memory is 75% full. If the entire file is loaded, the following message and prompt is displayed:

```
End of input file
*_
```

You can then edit the file.

> Note: If you have not used the /B parameter, EDLIN will stop loading the file when the first Ctrl-Z is encountered in the file's text. If you wish to edit a file that is known to contain embedded Ctrl-Z characters (end-of-file marks), you should use the /B parameter. EDLIN will then process the entire file regardless of any embedded end-of-file marks.

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

If the entire file cannot be loaded into memory, EDLIN loads lines until memory is 75% full, then displays the * prompt. You can then edit the portion of the file that is in memory.

To edit the remainder of the file, you must write some of the edited lines to disk in order to free memory so that you can load unedited lines from disk into memory. Refer to the Write Lines and Append Lines commands in this chapter for the procedure you will use.

Editing a New File

If the specified file does not exist on the drive, a new file is created with the specified name. The following message and prompt are displayed:

```
New file
*_
```

You can now create a new file by entering the desired lines of text. To begin entering text, you must enter an I command to insert lines.

When you have completed the editing session, you can save the original and updated (new) files by using the End Edit command. The End Edit command is discussed in this chapter in the section called "The EDLIN Commands." The original file is renamed to an extension of .BAK, and the new file has the filename and extension you specified in the EDLIN command.

> Note: You cannot edit a file with a filename extension of .BAK with EDLIN because the system assumes it is a backup file. If you find it necessary to edit such a file, rename the file to another extension; then start EDLIN and specify the new name.

The EDLIN Command Parameters

Parameter	Definition
line	Denotes when you must specify a line number. There are three possible entries that you can make using this parameter: 1. Enter a decimal integer from 1-65529. If you specify a number greater than the number of lines that are in memory, the line will be added after the last line that exists. Line numbers must be separated from each other by a comma or space. OR 2. Enter a pound sign (#) to specify the line after the last line in memory. Entering a # has the same effect as specifying a number greater than the number of lines in memory.

Parameter	Definition
line	OR 3. Enter a period (.) to specify the current line. The current line indicates the location of the last change to the file, but it is not necessarily the last line displayed. The current line is marked by an asterisk (*) between the line number and the first character of text in the line. For example: 10:*FIRST CHARACTER OF TEXT
n	Denotes when you must specify a number of lines. Enter the number of lines that you want to write to diskette or load from diskette. You only use this parameter with the Write Lines and Append Lines commands. These commands are meaningful only if the file to be edited is too large to fit in memory.
string	Denotes when you must enter one or more characters to represent text to be found, replaced, deleted, or to replace other text. You only use this parameter with the Search Text and Replace Text commands.

The EDLIN Commands

This section describes the EDLIN commands and tells how to use them. The commands are in alphabetical order; each with its purpose, format and remarks. Examples are provided where appropriate.

Information Common to All EDLIN Commands

The following information applies to all EDLIN commands:

- With the exception of the Edit Line command, all commands are a single letter.
- With the exception of the End Edit and Quit Edit commands, commands are usually preceded and/or followed by parameters.
- Enter commands and string parameters in uppercase or lowercase, or a combination of both.
- Separate commands and parameters with delimiters for readability; however, a delimiter is only required between two adjacent line numbers. Remember, delimiters are spaces or commas.
- Commands become effective only after you press the Enter key.
- Stop commands by pressing the Ctrl-Break keys.
- For commands producing a large amount of output, press Ctrl-Num Lock to suspend the display so that you can read it before it scrolls away. Press any other character to restart the display.

- Use the control keys and DOS editing keys, described in the *DOS User's Guide*, while using EDLIN. They are very useful for editing *within a line*, while the EDLIN commands can be used for editing operations on *entire lines*.

- The prompt from EDLIN is an asterisk (*).

- It is possible to refer to line numbers relative to the current line. Use a Minus (–) sign and a number to indicate a line before the current line. Use a Plus (+) sign and a number to indicate a line after the current line. For example:

  ```
  –10,+10L
  ```

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

- Multiple commands can be entered on one command line. When you enter the command to edit a single line using [*line*], you must use a semicolon to separate the commands on the line. In the case of the Search or Replace command the [*string*] can be terminated by Ctrl-Z (F6) instead of the Enter key. Otherwise, one command can follow another without any special delimiting characters. For example:

  ```
  15;–5,+5L
  ```

  edits line 15 and then displays lines 10 through 20 on the screen.

- Control characters can be inserted into the text, or can be used in the strings for the Search text and Replace text commands. To enter a control character, press Ctrl-V, then enter the desired control character in uppercase. For example, the sequence Ctrl-V, followed by Z generates the control character Ctrl-Z.

Append Lines Command

Purpose: Adds the specified number of lines from disk to the file being edited in memory. The lines are added at the end of the current lines in memory.

Format: [*n*]A

Remarks: This command is only meaningful if the file being edited is too large to fit in memory. As many lines as possible are read into memory for editing when you start EDLIN.

To edit the remainder of the file that will not fit into memory, you must write edited lines in memory to disk before you can load unedited lines from disk into memory by using the Append Lines command. Refer to the Write Lines command for information on how to write edited lines to disk.

Notes:

1. If you do not specify the number of lines, lines are appended to memory until available memory is 75% full. No action is taken if available memory is already 75% full.

2. The message **End of input file** is displayed when the Append Lines command has read the last line of the file into memory.

Copy Lines Command

Purpose: Copies the lines in the specified range to the line number specified by the third parameter. The new data is placed ahead of the line that was specified in the third parameter. This third parameter is not optional. The operation is repeated the number of times specified in *count*.

Format: [*line*],[*line*],*line*[,*count*]C

Remarks: The parameter *count* defaults to 1. To repeat text specify the number of times the operation is to be performed in *count*. If the first parameter or the second parameter is omitted, the default is the current line. This effectively copies the current line to the specified line. The file is renumbered accordingly. The first of the copied lines becomes the current line. For example:

```
1,5,8C
```

copies lines 1 through 5 to line 8. Line 8 becomes the current line.

> Note: The line numbers must not overlap or an error is reported. Also, the characters – and + are not allowed in the *count* field.

Delete Lines Command

Purpose: Deletes a specified range of lines.

Format: [*line*][,*line*]D

Remarks: The line following the deleted range becomes the current line, even if the deleted range includes the last line in memory. The current line and any following lines are renumbered.

Default values are supplied if either one or both of the parameters are omitted.

If you omit the first parameter, as in:

,*line*D

deletion starts with the current line and ends with the line specified by the second parameter. The beginning comma is required to indicate the omitted first parameter.

If you omit the second parameter, as in:

```
lineD
```

or

```
line,D
```

only the one specified line is deleted. If you omit both parameters, as in:

```
D
```

only the current line is deleted, and the line that follows becomes the current line.

Example: Assume that you want to edit the following file. The current line is line 29.

```
 1: This is a sample file
 2: used to demonstrate
 3: line deletion
 4: and dynamic
 5: line number generation.
 .
 .
 .
25: See what happens
26: to the lines
27: and line numbers
28: when lines are
29:*deleted.
```

If you want to delete a range of lines, from 5-25, enter:

```
5,25D
```

The result is:

```
1: This is a sample file
2: used to demonstrate
3: line deletion
4: and dynamic
5:*to the lines
6: and line numbers
7: when lines are
8: deleted.
```

Lines 5-25 are deleted from the file. Lines 26-29 are renumbered to 5-8. Line 5 becomes the current line. If you want to delete the current and the following line, enter:

```
,6D
```

The result is:

```
1: This is a sample file
2: used to demonstrate
3: line deletion
4: and dynamic
5:*when lines are
6: deleted.
```

Lines 5-6 are deleted from the file. Lines 7-8 are renumbered to 5-6. Line 5 is still the current line, but now it has different text.

If you want to delete a single line, say line 2, enter:

```
2D
```

The result is:

```
1: This is a sample file
2:*line deletion
3: and dynamic
4: when lines are
5: deleted.
```

Line 2 is deleted. Lines 3-6 are renumbered to 2-5. The new line 2 becomes the current line. If you want to delete only the current line, enter:

```
D
```

The result is:

```
1: This is a sample file
2:*and dynamic
3: when lines are
4: deleted.
```

The current line, line 2, is deleted. Lines 3-5 are renumbered to 2-4. The new line 2 becomes the current line.

Edit Line Command

Purpose: Allows you to edit a line of text. You must enter the line number of the line to be edited, or enter a period (.) to indicate the current line.

Format: [*line*]

Remarks: If you just press Enter, you specify that the line after the current line is to be edited.

The line number and its text are displayed and the line number is repeated on the line below.

You can use the control keys and the editing keys, described in the *DOS User's Guide*, to edit the line, or you can replace the entire line by typing new text.

When you press the Enter key, the edited line is placed in the file and becomes the current line.

If you decide not to save the changed line, press either Esc or Ctrl-Break instead of Enter. The original line remains unchanged. Pressing the Enter key with the cursor at the beginning of the line has the same effect as pressing Esc or Ctrl-Break.

If the cursor is in any position other than the beginning or the end of a line, pressing Enter erases the rest of the line.

Edit Line Command

Example: Assume that you want to edit line 6. The following display would appear on the screen:

```
*6
   6: This is a sample unedited line.
   6: _
```

The first line is your request to edit line 6, followed by the two-line display response.

If you want to move the cursor to the letter **u**, press F2 and enter:

```
u
```

The result is:

```
*6
   6: This is a sample unedited line.
   6: This is a sample _
```

If you want to delete the next two characters and keep the remainder of the line, press Del twice; then press F3.

The result is:

```
*6
   6: This is a sample unedited line.
   6: This is a sample edited line._
```

Now you can take one of the following actions:

- Press Enter to save the changed line.
- Extend the changed line by typing more text. You are automatically in insert mode when the cursor is at the end of a line.
- Press F5 to do additional editing to the changed line without changing the original line.
- Press Esc or Ctrl-Break to cancel the changes you made to the line. The original contents of the line will be preserved.

End Edit Command

Purpose: Ends EDLIN and saves the edited file.

Format: E

Remarks: The edited file is saved by writing it to the drive and filename specified when you started EDLIN.

The original file, the one specified when EDLIN was started, is given a .BAK filename extension. A .BAK file will not be created if there is no original file; that is, if you created a new file instead of updating an old file during the editing session.

EDLIN returns to the DOS command processor, which displays the command prompt.

Notes:

1. Be sure your disk has enough free space to save the entire file. If your disk does not have enough free space, only a portion of the file is saved. The portion in memory that is not written to disk is lost. In this case, your original file will not be renamed to .BAK, and the portion of data that was written to disk will have a filename extension of $$$.

2. EDLIN appends a carriage return, line feed sequence to the end of the file if they were not already present, to delimit the last line of text in the file. Also, a Ctrl-Z character is added as the last character in the saved file. This serves as an end-of-file mark.

Insert Lines Command

Purpose: Inserts lines of text immediately *before* the specified line. When you create a new file, you must enter the Insert Lines command before text can be inserted.

Format: [*line*]I

Remarks: If you do not specify line, or if you specify line as a period (.), the insert is made immediately before the current line.

If the line number you specify is greater than the highest existing line number, or if you specify # as the line number, the insertion is made after the last line in memory.

EDLIN displays the appropriate line number so that you can enter more lines, ending each line by pressing Enter. During the insert mode of operation, successive line numbers appear automatically each time Enter is pressed.

You must press Ctrl-Break to discontinue the insert mode of operation.

The line that follows the inserted lines becomes the current line, even if the inserted lines are added to the end of the lines in memory. The current line and any remaining lines are renumbered.

Example: Assume that you want to edit the following file. Line 3 is the current line:

```
1: This is a sample file
2: used to demonstrate
3:*line deletion
4: and dynamic
5: line number generation.
```

If you want to insert text before line 4, the entry and immediate response look like this:

```
*4I
   4:* _
```

Now, if you want to insert two new lines of text, enter:

```
*4 I
   4:*First new line of text
   5:*Second new line of text
   6:*
```

and press Ctrl-Break.

The original lines 4 and 5 are now renumbered to lines 6 and 7.

If you display the file with a List Lines command, the file looks like this:

```
1: This is a sample file
2: used to demonstrate
3: line deletion
4: First new line of text
5: Second new line of text
6:*and dynamic
7: line number generation.
```

If the two lines that were inserted had been placed at the beginning of the file, the screen would look like this:

```
1: First new line of text
2: Second new line of text
3:*This is a sample file
4: used to demonstrate
5: line deletion
6: and dynamic
7: line number generation.
```

If the two lines that were inserted had been placed immediately before the current line (3 I or . I or I), the screen would look like this:

```
1: This is a sample file
2: used to demonstrate
3: First new line of text
4: Second new line of text
5:*line deletion
6: and dynamic
7: line number generation.
```

If the two inserted lines had been placed at the end of the file (6 I or # I), the screen would look like this:

```
1: This is a sample file
2: used to demonstrate
3: line deletion
4: and dynamic
5: line number generation.
6: First new line of text
7: Second new line of text
```

List Lines Command

Purpose: Displays a specified range of lines.

The current line remains unchanged.

Format: [*line*][,*line*]L

Remarks: Default values are provided if either one or both of the parameters are omitted.

If you omit the first parameter, as in:

,*line*L

the display starts 11 lines before the current line and ends with the specified line. The beginning comma is required to indicate the omitted first parameter.

> Note: If the specified line is more than 11 lines before the current line, the display is the same as if you omitted both parameters. (An example is provided in this section showing both parameters omitted.)

If you omit the second parameter, as in:

```
lineL
```

or

```
line,L
```

a total of 23 lines are displayed, starting with the specified *line.*

If you omit both parameters, as in:

```
L
```

a total of 23 lines are displayed – the 11 lines before the current line, the current line, and the 11 lines after the current line. If there aren't 11 lines before the current line, then extra lines are displayed after the current line to make a total of 23 lines.

Example: Assume that you want to edit the following file. Line 15 is the current line.

```
 1: This is a sample file
 2: used to demonstrate
 3: line deletion
 4: and dynamic
 5: line number generation.
  .
  .
  .
15:*This is the current line (note the asterisk)
  .
  .
  .
25: See what happens
26: to the lines
27: and line numbers
28: when lines are
29: deleted.
```

If you want to display a range of lines, from 5-25, enter:

```
5,25L
```

The screen looks like this:

```
 5: line number generation.
  •
  •
  •
15:*This is the current line (note the asterisk)
  •
  •
  •
25: See what happens
```

If you want to display the first three lines, enter:

```
1,3L
```

The screen looks like this:

```
1: This is a sample file
2: used to demonstrate
3: line deletion
```

If you want to display 23 lines of the file, starting with line 3, enter:

```
3L
```

The screen looks like this:

```
 3: line deletion
 4: and dynamic
 5: line number generation.
  •
  •
  •
15:*This is the current line (note the asterisk)
  •
  •
  •
25: See what happens
```

If you want to display 23 lines centered around the current line, enter:

```
L
```

The screen looks like this:

```
 4: and dynamic
 5: line number generation.
  •
  •
  •
15:*This is the current line (note the asterisk)
  •
  •
  •
25: See what happens
26: to the lines
```

Move Lines Command

Purpose: Moves the range of lines specified by the first two *line* parameters ahead of the line specified in the third *line* parameter. The third parameter is not optional.

Format: [*line*],[*line*],*line*M

Remarks: Use this command to move a block of data from one location in the file to another. If the first or second line parameter is omitted, it will default to the current line. After the move, the first of the moved lines becomes the current line. The lines are renumbered according to the direction of the move. For example:

```
,+25,100M
```

moves the data from the current line plus 25 lines to line 100. If the arguments overlap an entry error is reported.

Page Command

Purpose: Lists the specified block of lines.

Format: [*line*][,*line*]P

Remarks: If the first *line* parameter is omitted, it defaults to the current line plus one. If the second *line* parameter is omitted, 23 lines are listed. The new current line becomes the last line displayed by the Page command and is marked with an asterisk. This command pages through a file displaying 23 lines at a time. It differs from the List Lines command in that it changes the current line.

Quit Edit Command

Purpose: Quits the editing session without saving any changes you may have entered.

Format: Q

Remarks: EDLIN prompts you to make sure you really don't want to save the changes.

Enter Y if you want to quit the editing session. No editing changes are saved and no .BAK file is created. Refer to the End Edit command for information about the .BAK file.

Enter N, or any other character, if you want to continue the editing session.

Example:

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

Replace Text Command

Purpose: Replaces all occurrences of the first string in the specified range of lines with the second string.

Notes:

1. If you omit the second string, Replace Text deletes all occurrences of the first string within the specified range of lines. If you omit both strings, EDLIN will reuse the search string entered with the most recent (previous) **S** or **R** command, and the Replace Text string entered with the last **R** command.

2. This command uses the F6 key as normally set up by DOS. If you have changed the meaning of the F6 key through "Extended Keyboard Control" (see Chapter 2 of the *DOS Technical Reference* manual), you should press Ctrl-Z where F6 is referred to below.

EDLIN displays the changed lines each time they are changed. The last line changed becomes the current line.

Format: [*line*][,*line*][?]R[*string*][<F6>*string*]

Remarks: You can specify the optional parameter ? to request a prompt (O.K.?) after each display of a modified line. Press Y or the Enter key if you want to keep the modification.

Enter any other character if you don't want the modification. In either case, the search continues for further occurrences of the first string within the range of lines, including multiple occurrences within the same line.

Defaults occur if either one or both of the *line* parameters are missing.

If you omit the first *line*, the search begins with the line after the current line. If you omit the second *line*, the search ends with the last line in memory. If you omit both *line* parameters, the system will search from the line following the current line to the last line in memory.

> Note: The first string begins with the character in the position immediately following the R, and continues until you press F6 or Ctrl-Z (or the Enter key if the second string is omitted).
>
> The second string begins immediately after you press F6 or Ctrl-Z and continues until you press Enter.

Example: Assume that you want to edit the following file. Line 7 is the current line.

```
1: This is a sample file
2: used to demonstrate
3: the Replace and Search Text commands.
4: This includes the
5: optional parameter ?
6: and required string
7:*parameter.
```

To replace all occurrences of **and** with or in the lines in memory, enter:

```
1,7 Rand
```

Then press F6, type or, and press Enter.

The result is:

```
3: The Replace or Search Text commands
6: or required string
```

Line 6 becomes the current line in the file, because line 6 was the last line changed. Notice that lines 1, 2, 4, 5, and 7 are not displayed because they were not changed.

Greater selectivity can be achieved by requesting a prompt (by using the ? parameter) after each display of a modified line. If you request a prompt, the screen looks like this:

```
*1,7?  Rand (Press F6, type or, and press Enter)
       3: the Replace or Search Text commands
O.K.?   Y
       3: the Replace or Search Text commands
O.K.?   N
       6: or required string
O.K.?   Y
*
```

Lines 3 and 6 are displayed like this:

```
3: the Replace or Search Text commands.
6: or required string
```

Search Text Command

Purpose: Searches a specified range of lines in order to locate a specified string.

Format: [*line*][,*line*][?]S[*string*]

Remarks: The first line to contain the specified string is displayed and the search ends (unless you use the ? parameter). The first line found that contains the specified string becomes the current line.

> Note: The Search command always searches for the exact same character in text. That is, it searches for UPPERCASE if you enter UPPERCASE, and lowercase if you enter lowercase.

You should specify the optional parameter ? if you would like a prompt (O.K.?) after each display of a line containing the specified string.

If you do not enter a string, the S command will use the last search string that was entered on a Replace or Search command. If the specified string is not found, the search ends and the message Not found is displayed. The current line remains unchanged. If you enter Y or press the Enter key, the line that matches the specified string becomes the current line and the search ends. Enter any other character to continue the search until another string is found, or until all lines within the range are searched. Once all the lines within the range are searched, the Not found message is displayed.

The system provides default values if you omit the first, second, or both line parameters. If you omit the first line parameter, the system defaults to the line following the current line. If you omit the second line parameter, the system defaults to the last line in memory. If you omit both line parameters, the system searches from the line following the current line to the last line in memory.

Notes:

1. The string begins with the character in the position immediately following the S and continues until you end the string by pressing the Enter key.

2. If you wish to place more than one command on a line containing a Search Text command, the Search Text command should end in a Ctrl-Z (F6), and the next command should begin in the following character position.

Example: Assume that you want to edit the following file. Line 7 is the current line.

```
1: This is a sample file
2: used to demonstrate
3: the Search Text command.
4: This includes the
5: optional parameter ?
6: and required string
7:*parameter.
```

Search Text Command

If you want to search for the first occurrence of **and** in the file, enter:

```
1,7 Sand
    or
1, Sand
    or
1Sand
```

The result is:

```
     3: the Search Text command.
*
```

The **and** is part of the word **command.** Notice that line 3 becomes the current line in the file.

Perhaps this is not the **and** you were looking for. To continue the search, simply enter the letter S and press Enter. The search will continue with the line following the current line (the line just found).

The screen looks like this:

```
*1,7 Sand
     3: the Search Text command.
*S
     6: and required string
*
```

Line 6 now becomes the current line in the file.

You can also search for strings by requesting a prompt (by means of the ? parameter) after each display of a matching line. In this case, the screen looks like this:

```
*1,7 ? Sand
     3: the Search Text command.
O.K.? N
     6: and required string
O.K.? Y
*
```

Transfer Lines Command

Purpose: Transfers (merges) the contents of a specified file into the file currently being edited.

Format: [*line*]T[*d*:]*filename*[*.ext*]

Remarks: The *filename* contents are inserted ahead of the *line* in the file being edited. If *line* is omitted, then the current line is used.

> **Note:** The file being merged is read from the current directory of the specified or default drive. If a path was specified when you issued the EDLIN command, then that path will be the current directory for that drive for the duration of the EDLIN session, and any Transfer Lines commands for that drive must be satisfied from the same directory.

Write Lines Command

Purpose: Writes a specified number of lines to diskette from the lines that are being edited in memory. Lines are written beginning with the line number 1.

Format: [*n*]W

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

To edit the remainder of your file, you must write edited lines in memory to diskette before you can load additional unedited lines from diskette into memory by using the Append Lines command.

> Note: If you do not specify the number of lines, lines are written until 25% of available memory is used. No action is taken if available memory is already less than 25% used. All lines are renumbered so that the first remaining line becomes number 1.

Summary of EDLIN Commands

The following chart is provided for quick reference.

Note: The section called "Format Notation" in Chapter 1 explains the notation used in the format of the following commands.

Command	Format
Append Lines	[*n*]A
Copy Lines	[*line*],[*line*],*line*,[*count*]C
Delete Lines	[*line*][,*line*]D
Edit Line	[*line*]
End Edit	E
Insert Lines	[*line*]I
List Lines	[*line*][,*line*]L
Quit Edit	Q
Move Lines	[*line*],[*line*],*line*M
Page	[*line*][,*line*]P
Replace Text	[*line*][,*line*][?]R[*string*][<F6>*string*]

Figure 3 (Part 1 of 2). EDLIN Commands

Command	Format
Search Text	[*line*][,*line*][?]S[*string*]
Transfer Lines	[*line*]T*filename*[.*ext*]
Write Lines	[*n*]W

Figure 3 (Part 2 of 2). EDLIN Commands

Chapter 7. The Linker (LINK) Program

Contents

Introduction

The linker (LINK) program is a program that:

- Combines separately produced object modules
- Searches library files for definitions of unresolved external references
- Resolves external cross-references
- Produces a printable listing that shows the resolution of external references and error messages
- Produces a relocatable load module

The LINK program resides on your DOS Supplemental Program Diskette. In this chapter, we show you how to use LINK. You should read all of this chapter before you start LINK.

Files

The linker processes the following input, output, and temporary files:

Input Files

Type	Default .ext	Override .ext	Produced by
Object	.OBJ	Yes	Compiler[1] or MACRO Assembler
Library	.LIB	Yes	Compiler
Automatic Response	(None)	N/A*	User

Figure 4. Input files used by the linker

*N/A – Not applicable.

[1] One of the optional compiler packages available for use with the IBM Personal Computer DOS.

Output Files

Type	Default .ext	Override .ext	Used by
Listing	.MAP	Yes	User
Run	.EXE	No	Relocatable loader (COMMAND. COM)

Figure 5. Output files used by the Linker

VM.TMP (Temporary File)

LINK uses as much memory as is available to hold the data that defines the load module being created. If the module is too large to be processed with the available amount of memory, the linker may need additional memory space. If this happens, a temporary file called VM.TMP is created on the DOS default drive.

When the overflow to the VM.TMP file has begun, the linker displays the following message:

```
VM.TMP has been created
Do not change diskette in drive x
```

If the VM.TMP file has been created on diskette, you should not remove the diskette until LINK ends. When LINK ends, it erases the VM.TMP file.

If the DOS default drive already has a file by the name of VM.TMP, it will be deleted by LINK and a new file will be allocated; the contents of the previous file are destroyed. Therefore, you should avoid using VM.TMP as one of your own filenames.

Definitions

Segment, *group*, and *class* are terms that appear in this chapter and in some of the messages in Appendix A. These terms describe the underlying function of LINK. An understanding of the concepts that define these terms provides a basic understanding of the way LINK works.

Segment

A *segment* is a contiguous area of memory up to 64K bytes in length. A segment may be located anywhere in memory on a *paragraph* (16-byte) boundary. Each of the four segment registers defines a segment. The segments can overlap. Each 16-bit address is an offset from the beginning of a segment. The contents of a segment are addressed by a segment register/offset pair.

The contents of various portions of the segment are determined when machine language is generated.

Neither size nor location is necessarily fixed by the compiler or assembler because this portion of the segment may be combined at link time with other portions forming a single segment.

A program's ultimate location in memory is determined at load time by the relocation loader facility provided in COMMAND.COM, based on whether you specify the /HIGH parameter. The /HIGH parameter is discussed later in this chapter.

Group

A *group* is a collection of segments that fit together within a 64K-byte segment of memory. The segments are named to the group by the assembler or compiler. A program may consist of one or more groups.

The group is used for addressing segments in memory. The various portions of segments within the group are addressed by a segment base pointer plus an offset. The linker checks that the object modules of a group meet the 64K-byte constraint.

Class

A *class* is a collection of segments. The naming of segments to a class affects the order and relative placement of segments in memory. The class name is specified by the assembler or compiler. All portions assigned to the same class name are loaded into memory contiguously.

The segments are ordered within a class in the order that the linker encounters the segments in the object files. One class precedes another in memory only if a segment for the first class precedes all segments for the second class in the input to LINK. Classes are not restricted in size. The classes are divided into groups for addressing.

Command Prompts

After you start the linker session, you receive a series of four prompts. You can respond to these prompts from the keyboard, respond to these prompts on the command line, or you can use a special diskette file called an *automatic response file* to respond to the prompts. An example of an automatic response file is provided in this chapter.

LINK prompts you for the names of the object, run, list, and library files. When the session is finished, LINK returns to DOS and the DOS prompt is displayed. If linking is unsuccessful, LINK displays a message.

The prompts are described in order of their appearance on the screen. Defaults are shown in square brackets ([ ]) after the prompt. In the response column of the table, square brackets indicate optional entries. **Object Modules** is the only prompt that requires a response from you.

Prompt	Responses
Object Modules [.OBJ]:	[*d*:][*path*\\]*filename*[.*ext*] [+.[*d*:][*path*\\]*filename*[.*ext*]]...
Run File [filename.EXT]:	[*d*:][*path*][*filename*[.*ext*]]
List File [NUL.MAP]:	[*d*:][*path*][*filename*[.*ext*]]
Libraries [.LIB]:	[*d*:][[*path*\\]*filename*[.*ext*]] [+.[*d*:][[*path*\\]*filename*[.*ext*]]]...

Notes:

1. If you enter a filename without specifying the drive, the default drive is assumed. If you enter a filename without specifying the path, the default path is assumed. The libraries prompt is an exception—the linker will look for the libraries on the default drive and if not found, look on the drive specified by the compiler.

2. You can end the linker session prior to its normal end by pressing Ctrl-Break.

Detailed Description of the Command Prompts

The following detailed descriptions contain information about the responses that you can enter to the prompts.

Object Modules [.OBJ]:

Enter one or more file locations for the object modules to be linked. Multiple file locations must be separated by single plus (+) signs or blanks. If the extension is omitted from any filename, LINK assumes the filename extension **.OBJ**. If an object module has a different filename extension, the extension must be specified. Object filenames can not begin with the @ symbol (@ is reserved for using an automatic response file).

LINK loads segments into classes in the order encountered.

If you specify an object module on a diskette drive, but LINK cannot locate the file, it displays the following prompt:

```
Cannot find file object module
change diskette <hit ENTER>
```

If you specify an object module on a non-removable media (like a fixed disk), the linker session will end with the following message:

```
Cannot find file object module
```

You should insert the diskette containing the requested module. This permits .OBJ files from several diskettes to be included. On a single-drive system, diskette exchanging can be done safely *only* if VM.TMP has *not* been opened. As explained in the discussion of the VM.TMP file earlier in this chapter, a message will indicate if VM.TMP has been opened.

IMPORTANT: If a VM.TMP file has been opened on a diskette, you should *not* remove the diskette containing the VM.TMP file. Remember, once a VM.TMP file is opened on a diskette, the diskette it resides on cannot be removed.

After a VM.TMP file has been opened, if you specified an object module on the same disk that VM.TMP is on and LINK cannot find it, the linker session ends with the message:

Cannot find file *object module*

Run File [filename.EXE]:

The file specification you enter is created to store the run (executable) file that results from the LINK session. All run files receive the filename extension .EXE, even if you specify another extension. If you specify another extension, your specified extension is ignored.

The default filename for the run file prompt is the first filename specified on the object module prompt.

You can specify just a drive letter, or a path on the run file prompt. This changes the place where the run file *filename*.EXE is placed.

List File [NUL.MAP]:

The linker list file is sometimes called the linker *map*.

The list file is not created unless you specifically request it. You can request it by overriding the default with a drive letter, path, or *filename*[*.ext*]. If you do not include a filename extension, the default extension .MAP is used. If you do not enter anything, the DOS reserved filename NULL specifies that no list file will be created.

The list file contains an entry for each segment in the input (object) modules. Each entry also shows the offset (addressing) in the run file.

You can specify just a drive letter or a path on the list file prompt. This changes the place where the list file is placed.

> Note: If the list file is allocated to a file on diskette, that diskette must not be removed until the LINK has ended.

If you specify an object module on the same diskette drive as the diskette drive to which the list file is allocated, and LINK cannot find the object module, the linker session ends with the message:

```
Cannot find file object module
```

To avoid generating the list file on a diskette, you can specify the display or printer as the list file device. For example:

```
List File [NUL.MAP]: CON
```

If you direct the output to your display, you can also print a copy of the output by pressing Ctrl-PrtSc.

Libraries [.LIB]:

You may either list the file locations for your libraries, or just press the Enter key. If you press the Enter key, LINK defaults to the library provided as part of the Compiler package.

The LINK program will look for the Compiler package library on the default drive. If it cannot find the library there, then it will look for the library on the drive specified by the Compiler package. For linking objects from just the MACRO Assembler, there is no automatic default library search.

If you answer the library prompt, you specify a list of drive letters and *[path]filename.ext* separated by plus signs (+) or spaces. You can enter from one to eight library file locations. Specifying a drive letter tells linker to look on that drive instead of the Compiler package supplied drive for all subsequent libraries on the library prompt. The automatically searched library file specifications are conceptually placed at the end of the response to the library prompt.

LINK searches the library files in the order in which they are listed to resolve external references. When LINK finds the module that defines the external symbol, the module is processed as another object module.

If two or more libraries have the same filename, regardless of the location, only the first library in the list is searched.

When LINK cannot find a library file, it displays a message like this:

```
Cannot find library A:library file
Enter new drive letter:
```

The drive that the indicated library is located on must be entered.

The following library prompt responses may be used:

Libraries [.LIB]: B:

Look for compiler .LIB on drive B.

Libraries [.LIB]: B:USERLIB

Look for USERLIB.LIB on drive B and compiler.LIB on drive A.

Libraries [.LIB]: A:LIB1+LIB2+B:LIB3+A:

Look for LIB1.LIB and LIB2.LIB on drive A, LIB3.LIB on drive B, and compiler.LIB on drive A.

Linker Parameters

At the end of any of the four linker prompts, you may specify one or more parameters that instruct the linker to do something differently. Only the / and first letter of any parameter are required.

/DSALLOCATION

The /DSALLOCATION (/D) parameter directs LINK to load all data defined to be in DGROUP at the *high end* of the group. If the /HIGH parameter is specified, (module loaded high), this allows any available storage below the specifically allocated area within DGROUP to be allocated dynamically by your application and still be addressable by the same data space pointer.

Note: The maximum amount of storage which can be dynamically allocated by the application is 64K (or the amount actually available) minus the allocated portion of DGROUP.

If the /DSALLOCATION parameter is not specified, LINK loads all data defined to be in the group whose group name is DGROUP at the *low end* of the group, beginning at an offset of 0. The only storage thus referenced by the data space pointer should be that specifically defined as residing in the group.

All other segments of any type in any GROUP other than DGROUP are loaded at the low end of their respective groups, as if the /DSALLOCATION parameter were not specified.

For certain compiler packages, /DSALLOCATION is automatically used.

/HIGH

The /HIGH (/H) parameter causes the loader to place the run image as high as possible in storage. If you specify the /HIGH parameter, you tell the linker to cause the loader to place the run file as high as possible without overlaying the transient portion of COMMAND.COM, which occupies the highest area of storage when loaded. If you do not specify the /HIGH parameter, the linker directs the loader to place the run file as low in memory as possible.

The /HIGH parameter is used with the /DSALLOCATION parameter.

/LINE

For certain IBM Personal Computer language processors, the /LINE (/L) parameter directs LINK to include the line numbers and addresses of the source statements in the input modules in the list file.

/MAP

The /MAP (/M) parameter directs LINK to list all public (global) symbols defined in the input modules. For each symbol, LINK lists its value and segment-offset location in the run file. The symbols are listed at the end of the list file.

/PAUSE

The /PAUSE (/P) parameter tells LINK to display a message to you as follows:

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

This message allows you to insert the diskette that is to contain the run file.

/STACK:size

The *size* entry is any positive decimal value up to 65536 bytes. This value is used to override the size of the stack that the MACRO Assembler or compiler has provided for the load module being created. If you specify a value greater than 0 but less than 512, the value 512 is used.

If you do not specify /STACK (/S), the original stack size provided by the MACRO Assembler or compiler is used.

If the size of the stack is too small, the results of executing the resulting load module are unpredictable.

At least one input (object) module must contain a stack allocation statement, unless you plan to use the EXE2BIN program. This is automatically provided by compilers. For the MACRO Assembler, the source must contain a SEGMENT command that has the combine type of STACK. If a stack allocation statement was not provided, LINK returns a Warning: No Stack statement message.

How to Start the Linker Program

Before You Begin

- Make sure the files you will be using for linking are on the appropriate disks.
- Make sure you have enough free space on your disks to contain your files and any generated data.

You can start the linker program by using one of three options:

Option 1 – Console Responses

From your keyboard, enter:

```
LINK
```

The linker is loaded into memory and displays a series of four prompts, one at a time, to which you must enter the requested responses. (Detailed descriptions of the responses that you can make to the prompts are discussed in this chapter.)

If you enter a wrong response, such as an incorrectly spelled filename, you must press Ctrl-Break to exit LINK, then restart LINK. If the response in error has been typed but you haven't pressed Enter yet, you may delete the wrong characters (on that line only).

An example of a linker session using the console response option is provided in this chapter in the section called "Example Linker Session."

As soon as you have entered the last filename, the linker begins to run. If the linker finds any errors, it displays the errors on the screen as well as in the listing file.

> **Note:** After any of these responses, before pressing Enter, you may continue the response with a comma and the answer to what would be the next prompt, without having to wait for that prompt. If you end any with the semicolon (;), the remaining responses are all assumed to be the default. Processing begins immediately with no further prompting.

Option 2 – Command Line

From your keyboard, enter:

LINK ***objlist,runfile,mapfile,liblist*** **[*parm*]...;**

objlist	is a list of object modules separated by spaces or plus signs (+).
runfile	is the name you want to give the run file.
mapfile	is the name you want to give the linker map.
liblist	is a list of the libraries to be used, separated by plus signs (+) or spaces.
parm	is an optional linker parameter. Each parameter must begin with a slash (/).

The linker is loaded and immediately performs the tasks indicated by the command line.

When you use this command line, the prompts described in Option 1 are not displayed if you specified an entry for all four files or if the command line ends with a semicolon.

If an incomplete list is given and no semicolon is used, the linker prompts for the remaining unspecified files.

Each prompt displays its default, which may be accepted by pressing the Enter key, or overridden with an explicit filename or device name. However, if an incomplete list is given and the command line is terminated with a final semicolon, the unspecified files default without further prompting. The *parm*s are never prompted for, but may be added to the end of the command line or to any file specification given in response to a prompt.

Certain variations of this command line are permitted.

Examples:

LINK module
: The object module is MODULE.OBJ. A prompt is given, showing the default of MODULE.EXE. After the response is entered, a prompt is given showing the default of NUL.MAP. After the response is given, a prompt is displayed showing the default extension of .LIB.

LINK module;
: If the semicolon is added, no further prompts are displayed. The object module of MODULE.OBJ is linked, the run file is put into MODULE.EXE, and no list file is produced.

LINK module,,;
: This is similar to the preceding example, except the list file is produced in MODULE.MAP.

LINK module,,

Using the same example, but without the semicolon, MODULE.OBJ is linked, and the run file is produced in MODULE.EXE, but a prompt is given with the default of MODULE.MAP.

LINK module,,NUL;

No list file is produced. The run file is in MODULE.EXE. No further prompts are displayed.

Option 3 – Automatic Responses

It is often convenient to save responses to the linker for use at a later time. This is especially useful when long lists of object modules need to be specified.

Before using this option, you must create the automatic response file. It contains several lines of text, each of which is the response to a linker prompt. These responses must be in the same order as the linker prompts that were discussed earlier in this chapter. If desired, a long response to the object module or libraries prompt may be contained on several lines by using a plus sign (+) to continue the same response onto the next line.

To specify an automatic response file, you enter a file specification preceded by an @ symbol in place of a prompt response or part of a prompt response. The prompt is answered by the contents of the diskette file. The file specification may not be a reserved DOS filename.

From your keyboard, enter:

```
LINK @[d:][path]filename1[ext]
```

Use of the filename extension is optional and may be any name. There is no default extension.

Use of this option permits the command that starts LINK to be entered from the keyboard or within a batch file without requiring any response from you.

Example

Automatic Response File – RESP1

```
MODA+MODB+MODC+
MODB+MODE+MODF
```

Automatic Response File – RESP2

```
runfile/p
printout
```

Command line

```
LINK @RESP1+mymod,@RESP2;
```

Notes:

1. The plus sign at the end of the first line in RESP1 causes the modules listed in the first two lines to be considered as the input object modules. After reading RESP1, the linker returns to the command line and sees +mymod, so it includes MYMOD.OBJ in the first list of object modules as well.

2. Each of the above lines ends when you press the Enter key.

Example Linker Session

This example shows you the type of information that is displayed during a linker session.

Once we enter:

```
b:link
```

in response to the DOS prompt, the system responds with the following messages and prompts, which we answer as shown:

```
IBM Personal Computer Linker
Version 2.10 (C)Copyright IBM Corp. 1981, 1982, 1983

Object Modules [.OBJ]: example
Run File [EXAMPLE.EXE]: /map
List File [NUL.MAP]: prn/line
Libraries [.LIB]:
```

Notes:

1. By specifying /map, we get both an alphabetic listing and a chronological listing of public symbols.

2. By responding **prn** to the list file prompt, we send our output to the printer.

3. By specifying the /LINE parameter, LINK gives us a listing of all line numbers for all modules. (The /LINE parameter can generate a large amount of output.)

4. By just pressing Enter in response to the libraries prompt, an automatic library search is performed.

Once LINK locates all libraries, the linker map displays a list of segments in the relative order of their appearance within the load module. The list looks like this:

```
Start    Stop     Length   Name           Class

00000H   00028H   0029H    MAINQQ         CODE
00030H   000F6H   00C7H    ENTXQQ         CODE
00100H   00100H   0000H    INIXQQ         CODE
00100H   038D3H   37D4H    FILVQQ__CODE   CODE
038D4H   04921H   104EH    FILUQQ__CODE   CODE
 •
 •
 •
074A0H   074A0H   0000H    HEAP           MEMORY
074A0H   074A0H   0000H    MEMORY         MEMORY
074A0H   0759FH   0100H    STACK          STACK
075A0H   07925H   0386H    DATA           DATA
07930H   082A9H   097AH    CONST          CONST
```

The information in the Start and Stop columns shows a 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 of where these segments are loaded. To find the absolute address of where a segment is actually loaded, you must determine where the segment listed as being at relative zero is actually loaded; then add the absolute address to the relative address shown in the linker map. The procedure you use to determine where relative zero is actually located is discussed in this chapter, in the section called "How to Determine the Absolute Address of a Segment."

Now, because we specified the /MAP parameter, the public symbols are displayed by name and by value. For example:

```
Address          Publics by Name

0492:0003H       ABSNQQ
06CD:029FH       ABSRQQ
0492:00A3H       ADDNQQ
06CD:0087H       ADDRQQ
0602:000FH       ALLHQQ
 •
 •
 •
0010:1BCEH       WT4VQQ
0010:1D7EH       WTFVQQ
0010:1887H       WTIVQQ
0010:19E2H       WTNVQQ
0010:11B2H       WTRVQQ

Address          Publics by Value

0000:0001H       MAIN
0000:0010H       ENTGQQ
0000:0010H       MAINQQ
0003:0000H       BEGXQQ
0003:0095H       ENDXQQ
 •
 •
 •
F82B:F31CH       CRCXQQ
F82B:F31EH       CRDXQQ
F82B:F322H       CESXQQ
F82B:F5B8H       FNSUQQ
F82B:F5E0H       OUTUQQ
```

The addresses of the public symbols are in the *segment:offset* format, showing the location relative to zero as the beginning of the load module. In some cases, an entry may look like this:

```
F8CC:EBE2H
```

This entry appears to be the address of a load module that is almost one megabyte in size. Actually, the area being referenced is relative to a segment base that is pointing to a segment below the relative zero beginning of the load module. This condition produces a pointer that has effectively gone negative. The memory map which follows illustrates this point.

When LINK has completed, the following message is displayed:

```
Program entry point at 0003:0000
```

How to Determine the Absolute Address of a Segment

The linker map displays a list of segments in the relative order of their appearance within the load module. The information displayed shows a 20-bit hex address of each segment relative to location zero. The addresses that are displayed are not the absolute addresses of where these segments are actually located. To determine where relative zero is actually located, we must use DEBUG. DEBUG is described in detail in Chapter 8.

Using DEBUG,

1. Load the application. Note the segment value in CS and the offset within that segment to the entry point as shown in IP. The last line of the linker map also describes this entry point, but uses relative values, not the absolute values shown by CS and IP.

2. Subtract the relative entry as shown at the end of the map listing from the CS:IP value. For example, let's say CS is at 05DC and IP is at zero.

   The linker map shows the entry point at 0100:0000. (0100 is a segment ID or paragraph number; 0000 is the offset into that segment.)

   In this example, relative zero is located at 04DC:0000, which is 04DC0 absolute.

If a program is loaded low, the relative zero location is located at the end of the Program Segment Prefix, in the location DS plus 100H.

Messages

All messages, except for the warning messages, cause the LINK session to end. Therefore, after you locate and correct a problem, you must rerun LINK.

Messages appear both in the list file and on the display unless you direct the list file to CON, in which case the display messages are suppressed.

All of the linker messages are included in Appendix A.

Chapter 8. The DEBUG Program

Contents

Introduction

This chapter explains how to use the DEBUG program.

The DEBUG program can be used to:

- Provide a controlled testing environment so you can monitor and control the execution of a program to be debugged. You can fix problems in your program directly, and then execute the program immediately to determine if the problems have been resolved. You do not need to reassemble a program to find out if your changes worked.
- Load, alter, or display any file.
- Execute *object files*. Object files are executable programs in machine language format.

How to Start the DEBUG Program

To start DEBUG, enter:

```
DEBUG [d:][path][filename[.ext]] [parm1] [parm2]
```

If you enter *filename*, the DEBUG program loads the specified file into memory. You may now enter commands to alter, display, or execute the contents of the specified file.

If you do *not* enter a filename, you must either work with the present memory contents, or load the required file into memory by using the Name and Load commands. Then you can enter commands to alter, display, or execute the memory contents.

The optional parameters, *parm1* and *parm2*, represent the optional parameters for the named *filespec*. For example,

```
DEBUG DISKCOMP.COM A: B:
```

In this command, the A: and B: are the parameters that DEBUG prepares for the DISKCOMP program.

When the DEBUG program starts, the registers and flags are set to the following values for the program being debugged:

- The segment registers (CS, DS, ES, and SS) are set to the bottom of free memory; that is, the first segment after the end of the DEBUG program.

- The Instruction Pointer (IP) is set to hex 0100.

- The Stack Pointer (SP) is set to the end of the segment, or the bottom of the transient portion of the program loader, whichever is lower. The segment size at offset 6 is reduced by hex 100 to allow for a stack of that size.

- The remaining registers (AX, BX, CX, DX, BP, SI, and DI) are set to zero. However, if you start the DEBUG program with a filespec, the CX register contains the length of the file in bytes. If the file is greater than 64K, the length is contained in registers BX and CX (the high portion in BX).

- The flags are set to their cleared values. (Refer to the Register command.)

- The default disk transfer address is set to hex 80 in the code segment.

All of available memory is allocated; therefore, any attempt by the loaded program to allocate memory will fail.

Notes:

1. If a file loaded by DEBUG has an extension of .EXE, DEBUG does the necessary relocation and sets the segment registers, stack pointer, and Instruction Pointer to the values defined in the file. The DS and ES registers, however, point to the Program Segment Prefix at the lowest available segment. The BX and CX registers contain the size of the program (smaller than the file size).

   The program is loaded at the high end of memory if the appropriate parameter was specified when the linker created the file. Refer to ".EXE File Structure and Loading" in Chapter 9 of the *DOS Technical Reference* manual for more information about loading .EXE files.

2. If a file loaded by DEBUG has an extension of .HEX, the file is assumed to be in Intel hex format and is converted to executable form while being loaded.

The DEBUG Command Parameters

Parameter	Definition

address	Enter a one- or two-part designation in one of the following formats:
	
	● An alphabetic segment register designation, plus an offset value, such as:
	`CS:0100`
	
	● A segment address, plus an offset value, such as:
	`4BA:0100`
	
	● An offset value only, such as:
	`100`
	(In this case, each command uses a default segment.)
	
	Notes:
	
	1. In the first two formats, the colon is required to separate the values.
	
	2. All numeric values are *hexadecimal* and may be entered as 1-4 characters.
	
	3. The memory locations specified in address must be valid; that is, they must actually exist. Unpredictable results will occur if an attempt is made to access a nonexistent memory location.

Parameter	Definition
byte	Enter a one or two character *hexadecimal* value.
drive	Enter a single digit (for example, 0 for drive A or 1 for drive B) to indicate which drive data is to be loaded from or written to. (Refer to the Load and Write commands.)
filespec	Enter a one- to three-part file specification consisting of a drive designation, filename, and filename extension. All three fields are optional. However, for the Name command to be meaningful, you should at least specify a drive designator or a filename. (Refer to the Name command.)
list	Enter one or more byte and/or string values. For example, `E CS:100 F3 'XYZ' 8D 4 "abcd"` has five items in the list (that is, three byte entries and two string entries having a total of 10 bytes).
portaddress	Enter a 1-4 character *hexadecimal* value to specify an 8- or 16-bit port address. (Refer to the Input and Output commands.)

Parameter	Definition

range	Enter either of the following formats to specify the lower and upper addresses of a range:
	
	● *address address*
	
	For example:
	
	`CS:100 110`
	
	Note: Only an offset value is allowed in the second address. The addresses must be separated by a space or comma.
	
	● *address* L *value*
	
	where *value* is the number of bytes in *hexadecimal* to be processed by the command. For example:
	
	`CS:100 L 11`
	
	Notes:
	
	1. The limit for *range* is hex 10000. To specify that *value* within four *hexadecimal* characters, enter 0000 (or 0).
	
	2. The memory locations specified in *range* must be valid; that is, they must actually exist. Unpredictable results will occur if an attempt is made to access a non-existent memory location.

registername	Refer to the Register command.

Parameter	Definition
sector sector	Enter 1-3 character *hexadecimal* values to specify: 1. The starting relative sector number 2. The number of sector numbers to be loaded or written In DEBUG, relative sectors are obtained by counting the sectors on the disk surface. The sector at track 0, sector 1, head 0 (the first sector on the disk) is relative sector 0. The numbering continues for each sector on that track and head, then continues with the first sector on the next head of the same track. When all sectors on all heads of the track have been counted, numbering continues with the first sector on head 0 of the next track. **Note:** This is a change from the sector mapping used by DOS Version 1.10. The maximum number of sectors that can be loaded or written with a single command is hex 80. A sector contains 512 bytes. (Refer to the Load and Write commands.)
string	Enter characters enclosed in quotation marks. The quotation marks can be either single (') or double ("). The ASCII values of the characters in the string are used as a list of byte values.

Parameter	Definition
string (cont.)	Within a string, the *opposite* set of quotation marks can be used freely as characters. However, if the *same* set of quotation marks (as the delimiters) must be used within the string, then the quotation marks must be doubled. The doubling does not appear in memory. For example: 1. ‘This “literal” is correct’ 2. ‘This ‘ ‘literal’ ’ is correct’ 3. ‘This ‘literal’ is not correct’ 4. ‘This ““literal”” is not correct’ 5. “This ‘literal’ is correct” 6. “This ““literal”” is correct” 7. “This “literal” is not correct” 8. “This ‘ ‘literal’ ’ is not correct” In the second and sixth cases above, the word *literal* is enclosed in one set of quotation marks in memory. In the fourth and eighth cases above, the word *literal* is not correct unless you really want it enclosed in two sets of quotation marks in memory.

Parameter	Definition
value	Enter a 1-4 character *hexadecimal* value to specify: ● The numbers to be added and subtracted (refer to the Hexarithmetic command), or ● The number of instructions to be executed by the Trace command, or ● The number of bytes a command should operate on. (Refer to the Dump, Fill, Move, Search, and Unassemble commands.)

The DEBUG Commands

This section presents a detailed description of how to use the commands to the DEBUG program. The commands appear in alphabetical order; each with its format and purpose. Examples are provided where appropriate.

Information Common to All DEBUG Commands

The following information applies to the DEBUG commands:

- A command is a single letter, usually followed by one or more parameters.
- Commands and parameters can be entered in uppercase or lowercase, or a combination of both.
- Commands and parameters may be separated by delimiters. Delimiters are only required, however, between two consecutive hexadecimal values. Thus, these commands are equivalent:

```
dcs:100 110
d cs:100 110
d,cs:100,110
```

- Press Ctrl-Break to end commands.
- Commands become effective only after you press the Enter key.

- For commands producing a large amount of output, you can press Ctrl-Num Lock to suspend the display to read it before it scrolls away. Press any other character to restart the display.

- You can use the control keys and the DOS editing keys, described in Chapter 2 of *DOS User's Guide* while using the DEBUG program.

- If a syntax error is encountered, the line is displayed with the error pointed out as follows:

  ```
  d cs:100 CS:110
            error
  ```

  In this case, the Dump command is expecting the second address to contain only a hexadecimal offset value. It finds the S, which is not a valid hexadecimal character.

- The prompt from the DEBUG program is a hyphen (-).

- The DEBUG program resides on your DOS Supplemental Program diskette.

Assemble Command

Purpose: To assemble IBM Personal Computer Macro Assembler language statements directly into memory.

Format: A[*address*]

Remarks: All numeric input to the Assemble command is in hexadecimal. The assembly statements you enter are assembled into memory at successive locations, starting with the address specified in *address*. If no address is specified, the statements are assembled into the area at CS:0100, if no previous Assemble command was used, or into the location following the last instruction assembled by a previous Assemble command. When all desired statements have been entered, press Enter when you are prompted for the next statement, to return to the DEBUG prompt.

DEBUG responds to invalid statements by displaying:

```
^Error
```

and redisplaying the current assemble address.

DEBUG supports standard 8086/8088 assembly language syntax (and the 8087 instruction set), with the following rules:

- All numeric values entered are hexadecimal and can be entered as 1-4 characters.

- Prefix mnemonics must be entered in front of the opcode to which they refer. They can also be entered on a separate line.

- The segment override mnemonics are CS:, DS:, ES:, and SS:.

- String manipulation mnemonics must explicitly state the string size. For example, MOVSW must be used to move word strings and MOVSB must be used to move byte strings.

- The mnemonic for the far return is RETF.

- The assembler will automatically assemble short, near, or far jumps and calls depending on byte displacement to the destination address. These can be overridden with the NEAR OR FAR prefix. For example:

  ```
  0100:0500 JMP 502       ; a 2 byte short jump
  0100:0502 JMP NEAR 505 ; a 3 byte near jump
  0100:0505 JMP FAR 50A  ; a 5 byte far jump
  ```

  The NEAR prefix can be abbreviated to NE, but the FAR prefix cannot be abbreviated.

- DEBUG cannot tell whether some operands refer to a word memory location or a byte memory location. In this case, the data type must be explicitly stated with the prefix "WORD PTR" or "BYTE PTR". DEBUG will also accept the abbreviations "WO" and "BY". For example:

```
NEG BYTE PTR [128]
DEC WO [SI]
```

- DEBUG also cannot tell whether an operand refers to a memory location or to an immediate operand. DEBUG uses the common convention that operands enclosed in square brackets refer to memory. For example:

```
MOV AX,21           ;Load AX with 21H
MOV AX,[21]         ;Load AX with the
                    contents of memory
                    location 21H
```

- Two popular pseudo-instructions have also been included. The DB opcode will assemble byte values directly into memory. The DW opcode will assemble word values directly into memory. For example:

```
DB   1,2,3,4,"THIS IS AN EXAMPLE"
DB   "THIS IS A QUOTE: " '
DB   "THIS IS A QUOTE: ' "

DW   1000,2000,3000", BACH"
```

- All forms of the register indirect commands are supported. For example:

```
ADD   BX,34[BP+2].[SI—1]
POP   [BP+DI]
PUSH  [SI]
```

- All opcode synonyms are supported. For example:

```
LOOPZ  100
LOOPE  100

JA     200
JNBE   200
```

- For 8087 opcodes the WAIT or FWAIT prefix must be explicitly specified. For example:

```
FWAIT FADD ST,ST(3)        ;This line will
                           assemble a
                           FWAIT prefix
 FLD TBYTE PTR [BX]        ;This line will not
```

Example:

```
C>debug
—a200
08B4:0200 xor ax,ax
08B4:0202 mov [bx],ax
08B4:0204 ret
08B4:0205
```

Compare Command

Purpose: Compares the contents of two blocks of memory.

Format: C *range address*

Remarks: The contents of the two blocks of memory are compared; the length of the comparison is determined from the *range*. If unequal bytes are found, their addresses and contents are displayed, in the form:

```
addr1   byte1   byte2   addr2
```

where, the first half (addr1 byte1) refers to the location and contents of the mismatching locations in *range*, and the second half (byte2 addr2) refers to the byte found in *address*.

If you enter only an offset for the beginning address of *range*, the C command assumes the segment contained in the DS register. To specify an ending address for *range*, enter it with only an offset value.

Example:

```
C 100L20 200
```

The 32 bytes of memory beginning at DS:100 are compared with the 32 bytes beginning at DS:200.

Dump Command

Purpose: Displays the contents of a portion of memory.

Format: D [*address*]

or

D [*range*]

Remarks: The dump is displayed in two parts:

1. A hexadecimal portion. Each byte is displayed in hexadecimal.

2. An ASCII portion. The bytes are displayed as ASCII characters. Unprintable characters are indicated by a period (.).

With a 40-column system display format, each line begins on an 8-byte boundary and shows 8 bytes.

With an 80-column system display format, each line begins on a 16-byte boundary and shows 16 bytes. There is a hyphen between the 8th and 9th bytes.

Note: The first line may have fewer than 8 or 16 bytes if the starting address of the dump is not on a boundary. In this case, the second line of the dump begins on a boundary.

Dump Command

The Dump command has two format options:

Option 1

Use this option to display the contents of hex 40 bytes (40-column mode) or hex 80 bytes (80-column mode). For example:

```
D address

  or

D
```

The contents are dumped starting with the specified address.

If you do not specify an address, the D command assumes the starting address is the location following the last location displayed by a previous D command. Thus, it is possible to dump consecutive 40-byte or 80-byte areas by entering consecutive D commands without parameters.

If no previous D command was entered, the location is offset hex 100 into the segment originally initialized in the segment registers by DEBUG.

> **Note:** If you enter only an offset for the starting address, the D command assumes the segment contained in the DS register.

Option 2

Use this option to display the contents of the specified address range. For example:

D *range*

Note: If you enter only an offset for the starting address, the D command assumes the segment contained in the DS register. If you specify an ending address, enter it with only an offset value.

For example:

```
D cs:100 10C
```

A 40-column display format might look like this:

```
04BA:0100  42 45 52 54 41 20 54 00
                          BERTA T.

04BA:0108  20 42 4F 52 47
                             BORG
```

Enter Command

Purpose: The Enter command has two modes of operation:

— Replaces the contents of one or more bytes, starting at the specified address, with the values contained in the list. (See Option 1.)

— Displays and allows modification of bytes in a sequential manner. (See Option 2.)

Format: E *address* [*list*]

Remarks: If you enter only an offset for the address, the E command assumes the segment contained in the DS register.

The Enter command has two format options:

Option 1

Use this option to place the list in memory beginning at the specified address.

E *address list*

For example:

```
E ds:100 F3 "xyz" 8D
```

Memory locations ds:100 through ds:104 are filled with the five bytes specified in the list.

Option 2

Use this option to display the address and the byte of a location, then the system waits for your input.

For example:

```
E address
```

Now you can take one of the following actions:

1. Enter a one or two character *hexadecimal* value to replace the contents of the byte; then take any of the next three actions:

2. Press the space bar to advance to the next address. Its contents are displayed. If you want to change the contents take action 1, above.

   To advance to the next byte without changing the current byte, press the space bar again.

3. Enter a hyphen (-) to back up to the preceding address. A new line is displayed with the preceding address and its contents. If you want to change the contents take action 1, above.

   To back up one more byte without changing the current byte, enter another hyphen.

4. Press the Enter key to end the Enter command.

   Note: Display lines can have 4 or 8 bytes of data, depending on whether the system display format is 40- or 80-column. Spacing beyond an 8-byte boundary causes a new display line, with the beginning address, to be started.

   For example:

   ```
   E cs:100
   ```

   might cause this display:

   ```
   04BA:0100  EB._
   ```

   To change the contents of 04BA:0100 from hex EB to hex 41, enter 41.

   ```
   04BA:0100  EB.41 _
   ```

   To see the contents of the next three locations, press the space bar three times. The screen might look like this:

   ```
   04BA:0100   EB.41   10.   00.   BC._
   ```

To change the contents of the current location (04BA:0103) from hex BC to hex 42, enter 42.

```
04BA:0100  EB.41   10.  00.  BC.42_
```

Now, suppose you want to back up and change the hex 10 to hex 6F. This is what the screen would look like after entering two hyphens and the replacement byte:

```
04BA:0100  EB.41   10.  00.  BC.42-
04BA:0102  00.-
04BA:0101  10.6F_
```

Press the Enter key to end the Enter command. You will see the hyphen (-) prompt.

Fill Command

Purpose: Fills the memory locations in the range with the values in the list.

Format: F *range list*

Remarks: If the list contains fewer bytes than the address range, the list is used repeatedly until all the designated memory locations are filled.

If the list contains more bytes than the address range, the extra list items are ignored.

> Note: If you enter only an offset for the starting address of the range, the Fill command assumes the segment contained in the DS register.

Example:

```
F 4BA:100 L 5 F3 "XYZ" 8D
```

Memory locations 04BA:100 through 04BA:104 are filled with the 5 bytes specified. Remember that the ASCII values of the list characters are stored. Thus, locations 100-104 will contain F3 58 59 5A 8D.

Go Command

Purpose: Executes the program you are debugging.

Stops the execution when the instruction at a specified address is reached (breakpoint), and displays the registers, flags, and the next instruction to be executed.

Format: G [=*address*] [*address* [*address*...]]

Remarks: Program execution begins with the current instruction, whose address is determined by the contents of the CS and IP registers, unless overridden by the =*address* parameter (the = must be entered). If =*address* is specified, program execution begins with CS:=*address*.

The Go command has two format options:

Option 1

Use this option to execute the program you are debugging without breakpoints. For example:

G [=*address*]

This option is useful when testing program execution with different parameters each time. (Refer to the Name command.) Be certain the CS:IP values are set properly before issuing the G command, if not using =*address*.

Option 2

This option performs the same function as Option 1 but, in addition, allows breakpoints to be set at the specified addresses. For example:

G [=*address*] *address*
  [*address...*]

This method causes execution to stop at a specified location so the system/program environment can be examined.

You can specify up to ten breakpoints in any order. You may wish to take advantage of this if your program has many paths, and you want to stop the execution no matter which path the program takes.

The DEBUG program replaces the instruction codes at the breakpoint addresses with an interrupt code (hex CC). If *any one* breakpoint is reached during execution, the execution is stopped, the registers and flags are displayed, and all the breakpoint addresses are restored to their original instruction codes. If no breakpoint is reached, the instructions are *not* restored.

Notes:

1. Once a program has reached completion (DEBUG has displayed the "Program terminated normally" message), it will be necessary to reload the program before it can be executed again.

2. Make sure that the address parameters refer to locations that contain valid 8088 instruction codes. If you specify an address that does not contain the first byte valid instruction, unpredictable results will occur.

3. The stack pointer must be valid and have 6 bytes available for the Go command; otherwise, unpredictable results will occur.

4. If only an offset is entered for a breakpoint, the G command assumes the segment contained in the CS register.

For example:

```
G 102 1EF 208
```

Execution begins with the current instruction, whose address is the current values of CS:IP. The =*address* parameter was not used.

Three breakpoints are specified; assume that the second is reached. Execution stops before the instruction at location CS:1EF is executed, the original instruction codes are restored, all three breakpoints are removed, the display occurs, and the Go command ends.

Refer to the Register command for a description of the display.

Hexarithmetic Command

Purpose: Adds the two hexadecimal values, then subtracts the second from the first.

Displays the sum and difference on one line.

Format: H *value value*

Example:

```
H 0F 8
17 07
```

The hexadecimal sum of 000F and 0008 is 0017, and their difference is 0007.

Input Command

Purpose: Inputs and displays (in hexadecimal) one byte from the specified port.

Format: I*portaddress*

Example:

```
I 2F8
6B
```

The single hexadecimal byte read from port 02F8 is displayed (6B).

Load Command

Purpose: Loads a file or absolute diskette sectors into memory.

Format: L [*address* [*drive sector sector*]]

Remarks: The maximum number of sectors that can be loaded with a single Load command is hex 80.

> **Note:** DEBUG displays a message if a diskette read error occurs. You can retry the read operation by pressing F3 to redisplay the Load command. Then, press the Enter key.

The Load command has two format options:

Option 1

Use this option to load data from the diskette specified by *drive*, and place the data in memory beginning at the specified *address*. For example:

L *address drive sector sector*

The data is read from the specified starting relative sector (first sector) and continues until the requested number of sectors is read (second sector).

> **Note:** If you only enter an offset for the beginning address, the L command assumes the segment contained in the CS register.

For example, to load data, you might enter:

```
L 4BA:100 1 0F 6D
```

The data is loaded from the diskette in drive B and placed in memory beginning at 4BA:100. Hex 6D (109) consecutive sectors of data are transferred, starting with relative sector hex 0F (15) (the 16th sector on the diskette).

Option 2

When issued without parameters, or with only the address parameter, use this option to load the file whose filespec is properly formatted in the file control block at CS:5C. For example:

```
L
```

or

L *address*

This condition is met by specifying the filespec when starting the DEBUG program, or by using the Name command.

> **Note:** If DEBUG is started with a filespec and subsequent Name commands are used, you may need to enter a new Name command for the proper filespec before issuing the Load command.

The file is loaded into memory beginning at CS:100 (or the location specified by *address*), and is read from the drive specified in the filespec (or from the default drive, if none was specified). Note that files with extensions of .COM or .EXE are always loaded at CS:100 – if you specify an address, it will be ignored.

The BX and CX registers are set to the number of bytes read; however, if the file being loaded has an extension of .EXE; BX and CX are set to the actual program size, and the file may be loaded at the high end of memory. Refer to the notes in "How to Start the DEBUG Program" at the beginning of this chapter for the conditions that are in effect when .EXE or .HEX files are loaded.

For example:

```
DEBUG
—N myprog
—L
—
```

The file named **myprog** is loaded from the default diskette and placed in memory beginning at location CS:0100.

Move Command

Purpose: Moves the contents of the memory locations specified by *range* to the locations beginning at the *address* specified.

Format: M *range address*

Remarks: Overlapping moves are always performed without loss of data during the transfer. (The source and destination areas share some of the same memory locations.)

The data in the source area remains unchanged unless overwritten by the move.

Notes:

1. If you enter only an offset for the beginning address of the range, the M command assumes the segment contained in the DS register. If you specify an ending address for the range, enter it with only an offset value.

2. If you enter only an offset for the address of the destination area, the M command assumes the segment contained in the DS register.

Example:

```
M CS:100 110 500
```

The 17 bytes of data from CS:100 through CS:110 are moved to the area of memory beginning at DS:500.

Name Command

Purpose: The Name command has two functions:

- Formats file control blocks for the first two filespecs, at CS:5C and CS:6C. (Starting DEBUG with a filespec also formats a file control block at CS:5C.)

  The file control blocks are set up for the use of the Load and Write commands, and to supply required filenames for the program being debugged.

- All specified filespecs and other parameters are placed exactly as entered, including delimiters, in a parameter save area at CS:81, with CS:80 containing the number of characters entered. Register AX is set to indicate the validity of the drive specifiers entered with the first two filespecs.

Format: N [*d*:][*path*]*filename*[.*ext*]

Remarks: If you start the DEBUG program without a filespec, you must use the Name command before a file can be loaded with the L command.

Example:

```
DEBUG
-N myprog
-L
-
```

To define filespecs or other parameters required by the program being debugged, enter:

```
DEBUG myprog
-N file1 file2
-
```

In this example, DEBUG loads the file **myprog** at CS:100, and leaves the file control block at CS:5C formatted with the same filespec. Then, the Name command formats file control blocks for *file1* and *file2* at CS:5C and CS:6C, respectively. The file control block for **myprog** is overwritten. The parameter area at CS:81 contains all characters entered after the N, including all delimiters, and CS:80 contains the count of those characters (hex 0C).

Output Command

Purpose: Sends the *byte* to the specified output port.

Format: O *portaddress byte*

Example: To send the byte value 4F to output port 2F8, enter:

```
O 2F8 4F
```

Quit Command

Purpose: Ends the DEBUG program.

Format: Q

Remarks: The file that you are working on in memory is *not* saved by the Quit command. You must use the Write command to save the file.

DEBUG returns to the command processor which then issues the normal command prompt.

Example:

```
-Q
A>
```

Register Command

Purpose: The Register command has three functions:

- It displays the hexadecimal contents of a single register, with the option of changing those contents.
- It displays the hexadecimal contents of all the registers, plus the alphabetic flag settings, and the next instruction to be executed.
- It displays the eight 2-letter alphabetic flag settings, with the option of changing any or all of them.

Format: R [*registername*]

Remarks: When the DEBUG program starts, the registers and flags are set to certain values for the program being debugged. (Refer to "How to Start the DEBUG Program" at the beginning of this chapter.)

Display a Single Register

The valid *registernames* are:

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

Both IP and PC refer to the instruction pointer.

For example, to display the contents of a single register, you might enter:

```
R AX
```

The system might respond with:

```
AX F1E4
:_
```

Now you may take one of two actions:

— Press Enter to leave the contents unchanged.

or

— Change the contents of the AX register by entering a 1-4 character hexadecimal value, such as hex FFF.

```
AX F1E4
:FFF_
```

Now pressing Enter changes the contents of the AX register to hex 0FFF.

Display All Registers and Flags

To display the contents of all registers and flags (and the next instruction to be executed), enter:

```
R
```

The system might respond with:

```
AX=0E00 BX=00FF CX=0007 DX=01FF
SP=039D BP=0000 SI=005C DI=0000
DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A NV UP DI NG NZ AC PE NC
04BA:011A  CD21          INT     21
```

The first four lines display the hexadecimal contents of the registers and the eight alphabetic flag settings. The last line indicates the location of the next instruction to be executed, and its hexadecimal and unassembled formats. This is the instruction pointed to by CS:IP.

Note: A system with an 80-column display shows:

1st line - 8 registers
2nd line - 5 registers and 8 flag settings
3rd line - next instruction information

A system with a 40-column display shows:

1st line - 4 registers
2nd line - 4 registers
3rd line - 4 registers
4th line - 1 register and 8 flag settings
5th line - next instruction information

Display All Flags

There are eight flags, each with 2-letter codes to indicate either a *set* condition or a *clear* condition.

The flags appear in displays in the same order as presented in the following table:

Flag Name	Set	Clear
Overflow (yes/no)	OV	NV
Direction (decrement/increment)	DN	UP
Interrupt (enable/disable)	EI	DI
Sign (negative/positive)	NG	PL
Zero (yes/no)	ZR	NZ
Auxiliary carry (yes/no)	AC	NA
Parity (even/odd)	PE	PO
Carry (yes/no)	CY	NC

Figure 6. Alphabetic Flag Settings

To display all flags, enter:

```
R F
```

If all the flags are in a *set* condition, the response is:

```
OV DN EI NG ZR AC PE CY-_
```

Now you can take one of two actions:

1. Press Enter to leave the settings unchanged.

2. Change any or all of the settings.

To change a flag, just enter its opposite code. The opposite codes can be entered in any order–with or without intervening spaces. For example, to change the first, third, fifth, and seventh flags, enter:

```
OV DN EI NG ZR AC PE CY - PONZDINV
```

They are entered in reverse order in this example.

Press Enter and the flags are modified as specified, the prompt appears, and you can enter the next command.

If you want to see if the new codes are in effect, enter:

```
R F
```

The response will be:

```
NV DN DI NG NZ AC PO CY -_
```

The first, third, fifth, and seventh flags are changed as requested. The second, fourth, sixth, and eighth flags are unchanged.

Note: A single flag can be changed only once per R F command.

Search Command

Purpose: Searches the *range* for the character(s) in the *list*.

Format: S *range list*

Remarks: All matches are indicated by displaying the addresses where matches are found.

A display of the prompt (-) without an address means that no match was found.

> **Note:** If you enter only an offset for the starting address of the range, the S command assumes the segment contained in the DS register.

Example: If you want to search the range of addresses from CS:100 through CS:110 for hex 41, enter:

```
S CS:100 110 41
```

If two matches are found the response might be:

```
04BA:0104
04BA:010D
```

If you want to search the same range of addresses as in the previous example for a match with the 4-byte-long list, enter:

```
S CS:100 L 11 41 "AB" E
```

The starting addresses of all matches are listed. If no match is found, no address is displayed.

Trace Command

Purpose: Executes one or more instructions starting with the instruction at CS:IP, or at =*address* if it is specified. The = must be entered. One instruction is assumed, but you can specify more than one with *value*. Displays the contents of all registers and flags *after each* instruction executes. For a description of the display format, refer to the Register command.

Format: T [=*address*][*value*]

Remarks: The display caused by the Trace command continues until *value* instructions are executed. Therefore, when tracing multiple instructions, remember you can suspend the scrolling at any time by pressing Ctrl-Num Lock. Resume scrolling by entering any other character.

Example:
```
T
```

If the IP register contains 011A, and that location contains B40E (MOV AH,0EH), this might be displayed:

```
AX=0E00  BX=00FF  CX=0007  DX=01FF
SP=039D  BP=0000  SI=005C  DI=0000
DS=04BA  ES=04BA  SS=04BA  CS=04BA
IP=011C NV UP DI NG NZ AC PE NC
04BA:011C  CD21       INT    21
```

This displays the results *after* the instruction at 011A is executed, and indicates the next instruction to be executed is the INT 21 at location 04BA:011C.

```
T 10
```

Sixteen instructions are executed (starting at CS:IP). The contents of all registers and flags are displayed after each instruction. The display stops after the 16th instruction has been executed. Displays may scroll off the screen unless you suspend the display by pressing the Ctrl-Num Lock keys.

Unassemble Command

Purpose: Unassembles instructions (translates the contents of memory into assembler-like statements) and displays their addresses and hexadecimal values, together with assembler-like statements. For example, a display might look like this:

```
04BA:0100  206472  AND [SI+72],AH
04BA:0103  FC      CLD
04BA:0104  7665    JBE 016B
```

Format: U [*address*]

or

U [*range*]

Remarks: The number of bytes unassembled depends on your system display format (whether 40 or 80 columns), and which option you use with the Unassemble command.

Notes:

1. In all cases, the number of bytes unassembled and displayed may be slightly more than either the amount requested or the default amount. This happens because the instructions are of variable lengths; therefore, to unassemble the last instruction may include more bytes than expected.

2. Make sure that the address parameters refer to locations containing valid 8088 instruction codes. If you specify an address that does not contain the first byte of a valid instruction, the display will be erroneous.

3. If you enter only an offset for the starting address, the U command assumes the segment contained in the CS register.

The Unassemble command has two format options:

Option 1

Use this option to either unassemble instructions without specifying an address, or to unassemble instructions beginning with a specified address. For example:

U

or

U ***address***

Sixteen bytes are unassembled with a 40-column display. "Thirty-two" or "display"; 32 bytes are unassembled with an 80-column display.

Instructions are unassembled beginning with the specified address.

If you do not specify an address, the U command assumes the starting address is the location following the last instruction unassembled by a previous U command. Thus, it is possible to unassemble consecutive locations, producing continuous unassembled displays, by entering consecutive U commands without parameters.

If no previous U command is entered, the location is offset hex 0100 into the segment originally initialized in the segment registers by DEBUG.

Option 2

Use this option to unassemble instructions in a specified address range. For example:

```
U range
```

All instructions in the specified address range are unassembled, regardless of the system display format.

> **Note:** If you specify an ending address, enter it with only an offset value.

For example:

```
U 04ba:0100 108
```

The display response might be:

```
04BA:0100  206472  AND [SI+72],AH
04BA:0103  FC      CLD
04BA:0104  7665    JBE 016B
04BA:0106  207370  AND [BP+DI+70],DH
```

The same display appears if you enter:

```
U 04BA:100 L 7
```

or

```
U 04BA:100 L 8
```

or

```
U 04BA:100 L 9
```

Write Command

Purpose: Writes the data being debugged to diskette.

Format: W [*address* [*drive sector sector*]]

Remarks: The maximum number of sectors that can be written with a single Write command is hex 80.

DEBUG displays a message if a diskette write error occurs. You can retry the write operation by pressing F3 to redisplay the Write command, then press the Enter key.

The Write command has two format options:

Option 1

Use this option to write data to diskette beginning at a specified address. For example:

```
W address drive sector sector
```

Write Command

The data beginning at the specified address is written to the diskette in the indicated drive. The data is written starting at the specified starting relative sector (first sector) and continues until the requested number of sectors are filled (second sector).

Notes:

1. Be extremely careful when you write data to absolute sectors because an erroneous sector specification will destroy whatever was on the diskette at that location.

2. If only an offset is entered for the beginning address, the W command assumes the segment contained in the CS register.

3. Remember, the starting sector and the sector count are both specified in *hexadecimal.*

For example:

```
W 1FD 1 100 A
```

The data beginning at CS:01FD is written to the diskette in drive B, starting at relative sector hex 100 (256) and continuing for hex 0A (10) sectors.

Option 2

This option allows you to use the Write command without specifying parameters or only specifying the address parameter. For example:

```
W
```

or

```
W address
```

When issued without parameters (or when issued with only the address parameter), the Write command writes the file (whose filespec is properly formatted in the file control block at CS:5C) to diskette.

This condition is met by specifying the filespec when starting the DEBUG program, or by using the Name command.

> Note: If DEBUG was started with a filespec and subsequent Name commands were used, you may need to enter a new Name command for the proper filespec before issuing the Write command.

In addition, the BX and CX registers must be set to the number of bytes to be written. They may have been set properly by the DEBUG or Load commands, but might have been changed by a Go or Trace command. You must be certain the BX and CX registers contain the correct values.

The file beginning at CS:100, or at the location specified by *address*, is written to the diskette in the drive specified in filespec or the default drive if none was specified.

The debugged file is written over the original file that was loaded into memory, or into a new file if the filename in the FCB didn't previously exist.

> **Note:** An error message is issued if you try to write a file with an extension of .EXE or .HEX. These files must be written in a specific format that DEBUG cannot support.

If you find it necessary to modify a file with an extension of .EXE or .HEX, and the exact locations to be modified are known, use the following procedure:

1. RENAME the file to an extension other than .EXE or .HEX.

2. Load the file into memory using the DEBUG or Load command.

3. Modify the file as needed in memory, but do not try to execute it with the Go or Trace commands. Unpredictable results would occur.

4. Write the file back using the Write command.

5. RENAME the file back to its correct name.

Summary of DEBUG Commands

The following chart is provided for quick reference.

The section called "Format Notation" in Chapter 1 explains the notation used in the format of the following commands.

Command	Purpose	Format
Assemble	Assembles statements	A [*address*]
Compare	Compares memory	C *range address*
Dump	Displays memory	D [*address*] or D [*range*]
Enter	Changes memory	E *address* [*list*]
Fill	Changes memory blocks	F *range list*
Go	Executes with optional breakpoints	G [=*address*] [*address* [*address*...]]
Hexarithmetic	Hexadecimal add-subtract	H *value value*
Input	Reads/displays input byte	I *portaddress*
Load	Loads file or absolute diskette sectors	L [*address [drive sector sector*]]
Move	Moves memory block	M *range address*

Figure 7 (Part 1 of 2). DEBUG Commands

Command	Purpose	Format
Name	Defines files and parameters	[*d*:] [*path*]*filename*[.*ext*]
Output	Sends output byte	O *portaddress byte*
Quit	Ends DEBUG program	Q
Register	Displays registers/flags	R [*registername*]
Search	Searches for characters	S *range list*
Trace	Executes and displays	T [=*address*] [*value*]
Unassemble	Unassembles instructions	U [*address*] or U [*range*]
Write	Writes file or absolute diskette sectors	W [*address*[*drive sector sector*]]

Figure 7 (Part 2 of 2). DEBUG Commands

Appendixes

Contents

Appendix A. Messages

Introduction

This chapter contains *device errors* (the message that DOS uses to indicate errors while reading or writing to devices on your system), and *Other messages* (the remainder of the DOS messages) in alphabetical order. Messages are listed in **bold** type and the explanation and action follow the message.

The first word of the description of each message is the name of the program or command that generated the message.

In some cases, the message could be generated by several different programs or commands. In that case, the first word is *commands*. Where the message is generated by an internal DOS function, the first word is *DOS*.

Responses

When any of the device error messages are displayed, the system waits for you to respond. If you know what caused the problem, you can take corrective action before you actually choose a response. The system waits until you make *one* of these responses. To recover from an error condition, the responses should be made in the following order:

R	**Retry** the operation because the error may not occur again. The system tries the disk read or write operation again. We strongly recommend that you use **Retry** first.
A	**Abort** the program. The system ends the program that requested the disk read or write.
I	**Ignore** the error condition and continue the program. (Be careful when choosing this response because data may be lost.) The system pretends the error did not occur and continues the program.

Device Error Messages

When an error is detected during reading or writing to any of the devices (disk drives, printer, etc.) on your system, DOS displays a message in the following format:

<type> **error reading** *<device>*
Abort, Retry, Ignore?
or
<type> **error writing** *<device>*
Abort, Retry, Ignore?

In these messages, *<device>* is the name of the device in error, such as **PRN**, or **B:**, and *<type>* is one of the types listed on the following pages:

Bad call format

Explanation: A device driver was passed an incorrect length request header.

Action: Refer to "Responses" at the beginning of this group of device messages.

- Use DEBUG.
- Review your programming specifications. Patch and reassemble.
- If you are using a purchased program, contact the dealer you purchased the device driver from.

Bad command

Explanation: A device driver has issued an invalid command to <*device*>.

Action: Refer to "Responses" at the beginning of this group of device messages.

- Review your device interface specification and DOS driver implementation to make sure everything you are trying to do is supported.
- Check your program to see if you have a coding problem that needs debugging.

Bad unit

Explanation: A device driver has been passed an invalid subunit number.

Action: Refer to "Responses" at the beginning of this group of device messages. If you are using a purchased program, contact the dealer you purchased the device driver from.

Data

Explanation: DOS was unable to read or write the data correctly. This message usually means a disk has developed a defective spot.

Action: Refer to "Responses" at the beginning of this group of device messages.

Disk error reading

Explanation: An error of a type not described elsewhere in this list has occurred.

Action: Refer to "Responses" at the beginning of this group of device messages. Choose **Retry** first. Then choose **Abort** if this problem requires further investigation by a programmer.

If you are using a purchased program, contact the dealer you purchased it from.

Disk error writing

Explanation: An error of a type not described elsewhere in this list has occurred.

Action: Refer to "Responses" at the beginning of this group of device messages. Choose **Retry** first. Then choose **Abort** if this problem requires further investigation by a programmer.

If you are using a purchased program, contact the dealer you purchased it from.

No paper

Explanation: The indicated printer is either out of paper or not turned on.

Action: Turn the printer ON, press the ONLINE switch, or add paper and retry.

Refer to "Responses" at the beginning of this group of device messages.

Non-DOS disk

Explanation: The file allocation table contains invalid information. The disk needs to be reformatted.

Action: Refer to "Responses" at the beginning of this group of device messages.

Try running CHKDSK to see if any corrective action is possible. Reformatting will correct the disk, but the files are lost forever.

Not ready

Explanation: The named device is not ready to accept or transmit data.

Action: Check that the disk drive door is closed and choose **Retry** for your response if this is the problem.

Refer to "Responses" at the beginning of this group of device messages.

Read fault

Explanation: DOS was unable to read the data from the device.

Action: Refer to "Responses" at the beginning of this group of device messages.

- Make sure the diskette is properly inserted in the drive.
- If you get the same message, choose **Abort** and rerun the command with a different disk.

Sector not found

Explanation: The sector containing the data could not be located on the disk.

Action: Refer to "Responses" at the beginning of this group of device messages.

If you get the same message, choose **Abort** and rerun the command with a different disk.

Seek

Explanation: The fixed disk or diskette drive was unable to locate the proper track on the disk.

Action:

- Make sure the diskette is properly inserted in the drive.
- Try a different drive
- Run CHKDSK

Refer to "Responses" at the beginning of this group of device messages.

Write fault

Explanation: DOS was unable to write the data to the device.

Action:

- Make sure the diskette is properly inserted in the drive.
- If the diskette is not the problem, choose **Retry**.
- If you get the same message, choose **Abort** and rerun the command with a different disk.

Refer to "Responses" at the beginning of this group of device messages.

Write protect

Explanation: An attempt was made to write on a write protected diskette.

Action: Investigate carefully before you decide to write on a write protected diskette.

> **Note:** One of the preceding messages will appear if you attempt to use a dual-sided diskette in a single-sided drive, or if you attempt to use a 9-sector-per-track diskette on a pre-version 2.00 or 2.10 level of DOS.
>
> **Warning:** If any of the preceding messages appear for a diskette drive, DO NOT change diskettes before responding with Abort, Retry, or Ignore.

Other Messages

The following messages are in alphabetical order.

A

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

> **Explanation:** LINK. Informational message. This message is displayed when you specify the /PAUSE parameter.
>
> **Action:** Insert your Runfile diskette into the appropriate drive and press Enter.

Access denied

> **Explanation:** DEBUG. An attempt was made to write to a file that is marked read only. The filename you are using is protected.
>
> **Action:** Use a different filename.

All files canceled by operator

> **Explanation:** PRINT/T. Informational message. You used the PRINT command with the /T parameter to cancel the printing of all queued files.
>
> **Action:** No action required. This message appears on the printer.

All specified file(s) are contiguous

Explanation: CHKDSK. Informational message. The file or files you named are all written sequentially on the disk.

Action: No action required.

Allocation error for file, size adjusted

Explanation: CHKDSK or CHKDSK/F. A filename precedes this message. An invalid cluster number was found in the file allocation table.

Action:

- If you specified the /F parameter, the file is truncated at the end of the last valid cluster.

- If you did not enter the /F parameter, the message is for your information and no action is needed. Enter:

  ```
  CHKDSK/F
  ```

  to correct the file size.

Amount read less than size in header

Explanation: EXE2BIN. The program portion of the file was smaller than indicated in the file's header.

Action: Recompile or reassemble the program, and then reLINK it.

An internal failure has occurred

Explanation: LINK. An error occurred in the linker program.

Action: Write down the conditions under which the message appeared. Report this to your Authorized IBM Personal Computer Dealer.

Attempt to access data outside of segment bounds

Explanation: LINK. An object file is invalid.

Action:

- Review the .ASM file or assembled listing for segmentation violations.
- Look for a bad reference or invalid instruction.

Attempted write protect violation

Explanation: FORMAT. The diskette being formatted is write protected and cannot be written on.

Action: In response to the displayed prompt, insert a new diskette and press any key to restart formatting.

B

Backup file sequence error

Explanation: RESTORE. A file to be restored was backed up on more than one diskette. You did not insert the diskette with the first part of the file.

Action: Rerun RESTORE and start with the correct diskette.

Bad command or filename

Explanation: DOS. The command you just entered is not a valid DOS command.

Action: Check the spelling of the command and re-enter it.

If the command name is spelled correctly, check to see that the default or specified drive contains the external command or batch file you are trying to execute.

Bad numeric parameter

Explanation: LINK. The value you specified with the /STACK parameter is not a valid numeric constant.

Action: Use a valid numeric constant and try LINK again.

Bad or missing Command Interpreter

Explanation: DOS. The disk that DOS is being started from does not contain a copy of COMMAND.COM, or an error occurred while the disk was being loaded.

This message also appears if COMMAND.COM has been removed from the directory it was in originally when DOS was started; or if the **COMSPEC=** parameter in the environment points to a directory not containing COMMAND.COM and DOS is trying to reload the command processor.

Action: Do a System Reset. If System Reset fails to solve the problem, start DOS with your backup DOS diskette. Then copy COMMAND.COM from the backup diskette to the root directory of the disk that failed.

Bad or missing <filename>

Explanation: DOS. This message appears only at startup and indicates one of the following:

a. The name of a device driver named in a **DEVICE=** <filename> parameter in the CONFIG.SYS file was not found.

b. A break address was out of bounds for the machine size.

c. An error occurred while the driver was being loaded. That driver is not installed by DOS.

Action:

a. For an incorrect device driver name, use the correct spelling for the device drive name, or use the diskette with the *named* device driver.

b. For items B and C, correct the coding in the device driver.

c. If you still cannot correct the problem, see your IBM Personal Computer dealer.

Batch file missing

Explanation: DOS. DOS could not locate the batch file it was processing. The file may have been erased or renamed by one of the steps within it. Batch processing stops and the DOS prompt appears.

Action:

1. If the filename was changed or if the file was deleted, correct the command that changed the name(s).

2. If the file was erased, use your backup copy. If you used EDLIN to create the file or make changes, rename the .BAK file to .BAT. Correct the command that deleted the file.

BF

Explanation: DEBUG. Bad flag. An invalid flag code setting was specified.

Action: Try the Register (R F) command again with the correct code.

BP

Explanation: DEBUG. Breakpoints. More than ten breakpoints were specified for the GO command.

Action: Re-enter the GO (G) command again with ten or fewer breakpoints.

BR

Explanation: DEBUG. Bad register. An invalid register name was specified.

Action: Re-enter the Register (R) command using a correct register name.

BREAK is On|Off

Explanation: BREAK. This message indicates the status of BREAK, either on or off.

Action: Enter the command you want. For example, if **Break is off** is displayed and **Break is on** is desired, enter the command:

```
BREAK ON
```

C

Cannot do binary reads from a device

Explanation: COPY. You used the **/B** parameter with a device name while trying to copy from the device. The copy cannot be performed in binary mode because COPY must be able to detect end-of-file from the device.

Action: Re-enter COPY and omit the **/B** parameter or use the **/A** parameter after the device name.

Cannot edit .BAK file—rename file

Explanation: EDLIN. Files with the extension .BAK are considered to be backup files, with more up-to-date versions of the files assumed to exist. Therefore, .BAK files shouldn't ordinarily be edited.

Action: If it is necessary to edit the .BAK file, rename the file giving it an extension other than .BAK. Or else, copy the file and give the copy a different filename extension.

Cannot find file *object file*
Change diskette <hit ENTER>

Explanation: LINK. The linker could not locate the specified object module on the drive.

Action: Insert the correct diskette with the *object file* on it and press Enter.

Warning: If it is necessary to change the diskette that contains the open VM.TMP file, you need to exit LINK by pressing Ctrl-Break instead of changing diskettes, and then restart LINK with a different drive specified to locate the object file. Otherwise there may be a loss of data on the diskette inserted. This usually occurs when using default drive values for the object file.

Cannot find library *library file*
Enter new drive letter:

Explanation: LINK. The specified library could not be found on the drive.

Action: Enter the correct letter for the drive the library is on.

Warning: If it is necessary to change the diskette that contains the open VM.TMP file, you need to exit LINK by pressing Ctrl-Break instead of different diskettes, and then restart LINK with a changing drive specified to locate the object file. Otherwise there may be a loss of data on the diskette inserted. This usually occurs when using default drive values for the object file.

Cannot format an ASSIGNed drive

Explanation: FORMAT. The ASSIGN *x=y* was executed prior to the attempted execution of FORMAT.

Action: Execute ASSIGN to restore the original drive letter assignment. Then proceed to execute FORMAT.

Cannot load COMMAND, system halted

Explanation: DOS. DOS attempted to reload the command processor, but the area in which DOS keeps track of available memory was destroyed; or the command processor was not found in the path specified by the COMSPEC parameter.

Action: Restart DOS. Don't forget to put the DOS diskette in the drive.

Cannot nest response file

Explanation: LINK. You used *@filespec* within an automatic response file. Automatic response files cannot be nested.

Action:

1. Change the initial auto response file to eliminate nested auto response file.

2. Or, if you did not intend this to be an auto response file, fix the syntax error.

Cannot open list file

Explanation: LINK. The directory or disk is full.

Action: Insert another disk, or delete some files from the disk that is full.

Cannot open overlay

Explanation: LINK. The directory or disk is full.

Action: Insert another disk, or delete some files from the disk that is full.

Cannot open response file

Explanation: LINK. The automatic response file could not be found.

Action: Include drive specifier and/or path for the response file. Place file on proper disk.

Cannot open temporary file

Explanation: LINK. The directory or disk is full.

Action: Insert another disk, or delete some files from the disk that is full.

Cannot start COMMAND, exiting

Explanation: DOS. While DOS was attempting to load another copy of the command processor, either the FILES= parameter in the configuration file was found to contain too small a value, or there is not enough available memory to contain the new copy of COMMAND.COM.

Action:

- Restart DOS.
- If necessary, increase the parameter value of FILES= parameter in CONFIG.SYS.

COMn: bbbb,p,d,s,t initialized

Explanation: MODE. Informational message. The Asynchronous Communications Adapter is initialized. The values represent:

n	adapter (COM1 or COM2)
bbbb	baud rate
p	parity

e	even
o	odd
n	none

s	stop bits (1 or 2)

t	type of serial device	
	p	serial printer (serial timeouts will be retried)
	–	other serial device (serial timeouts will not be retried)

Action: No action required. The feedback message from MODE shows its interpretation of the MODE command and the parameters you entered.

Compare error at offset XXXXXXXX

Explanation: COMP. Informational message. The files being compared contain different values at the displayed offset (in hexadecimal) into the file. The differing values are also displayed in hexadecimal.

Action: No action required. This message is for your information to give you the location that contains mismatching information in two files.

Compare error(s) on
Track xx, side xx

Explanation: DISKCOMP. One or more locations on the indicated track and side contain different information when the diskettes are compared.

Action: This message is to inform you that there is a difference between diskettes. If you want an exact copy of a diskette, use DISKCOPY.

Compare more diskettes (Y/N)?

Explanation: DISKCOMP.

- This message indicates completion of DISKCOMP.
- You may compare more than one set of diskettes without re-entering the DISKCOMP command.

Action: If you wish to compare another pair of diskettes, enter **Y**, and DISKCOMP will ask you to insert the required diskettes. If you do not want to compare any more diskettes, enter **N**.

Compare more files (Y/N)?

Explanation: COMP. COMP has finished comparing files. You may compare more files without re-entering the COMP command.

Action:

- If you wish to compare the contents of two more files, enter **Y**, and COMP will prompt you for the names of the files to compare.
- If you do not wish to compare more files, enter **N**.

Comparing x sectors per track, n side(s)

Explanation: DISKCOMP. The **n** will be either 1 or 2, indicating the number of sides that DISKCOMP will compare on the two diskettes. This number is determined by the number of sides DISKCOMP was able to successfully read from the first track of the first diskette. The *x* indicates the number of sectors per track found on the first diskette (8 or 9). If you use /8, then the number 8 will appear.

Action: If x or **n** is not what you expected let DISKCOMP finish comparing diskettes. Re-enter the DISKCOMP command with or without additional parameters.

Contains invalid cluster, file truncated

Explanation: CHKDSK. The file name preceding this message means that the file contains an invalid pointer to the data area.

Action: Use the /F parameter to truncate the file at the last valid data block. No corrective action occurs if CHKDSK is executed without the /F parameter.

Contains xxx non-contiguous blocks

Explanation: CHKDSK. Informational message. The file name preceding this message means that the file is not written sequentially on the disk—it is written in xxx pieces on different areas of the disk.

This message is for your information and does not indicate a problem with the disk.

Action: Since fragmented files take longer to read, you should consider copying badly fragmented files to another disk with the COPY command. This will record the file sequentially, resulting in better system performance when the file is read.

Convert directory to file (Y/N)?

Explanation: CHKDSK. The directory name preceding this message means the directory contains too much invalid information to be usable as a directory.

Action:

- If you reply **Y**, CHKDSK will convert the directory to a file so that you may examine it with DEBUG.

- If you reply **N**, the entry is not changed.

xxx lost clusters found in yyy chains.
Convert lost chains to files (Y/N)?

Explanation: CHKDSK located *xxx* blocks of the data area which were marked as allocated, but were not associated with a file. These clusters are assumed to contain "lost" data, and CHKDSK will ask whether you wish to free them, or to recover each chain into a separate file.

Action:

- If you reply Y and you have used the /F parameter, CHKDSK will recover each chain into a separate file.
- If you reply N, CHKDSK frees the blocks up so they can be allocated to new files.
- If CHKDSK was specified (no /F), then messages displayed afterwards are informational (no corrective action was taken).

Copy another (Y/N)?

Explanation: DISKCOPY. This message allows you to make exact images of additional diskettes without re-entering the DISKCOPY command.

Action:

- If you wish to copy another entire diskette, enter Y. DISKCOPY will ask you to insert the required diskettes.
- If you do not wish to make any more copies, enter N.

Copy complete

Explanation: DISKCOPY. Informational message.

Action: No action required. The source diskette contents have been successfully copied to the target diskette.

Copying x sectors per track, n side(s)

Explanation: DISKCOPY. The n will be either 1 or 2, indicating the number of sides that DISKCOPY has successfully read from the first track of the source diskette. The *x* will be 8 or 9, indicating the number of sectors per track found on the source diskette. If the diskette has been formatted double sided and subsequently formatted single sided, DISKCOPY will say it is copying *2 side(s)*. In this situation, use DISKCOPY /1.

If the diskette has ever been formatted with 9 sectors per track, then subsequently formatted for 8-sector-per-track use, DISKCOPY will say it is copying 9 sectors per track. However, DOS will only use the first 8 sectors per track to contain data.

Action: No action required.

D

DF

Explanation: DEBUG. Double flag. Conflicting codes were specified for a single flag.

Action: DEBUG is informing you that a flag can be changed only once per Register (R F) command.

Disk boot failure

Explanation: DOS. An error occurred when you tried to load DOS into memory.

Action: Restart the system. If subsequent attempts to start the system also fail, place a backup DOS diskette in drive A and restart your system.

Disk error writing FAT X

Explanation: CHKDSK. A disk error was encountered while CHKDSK was attempting to update the file allocation table (FAT) on the specified drive. X will be 1 or 2, depending on which of the 2 copies of the file allocation table could not be written.

Action: If this message appears twice, for FAT's 1 and 2, format the disk to make it usable again. If FORMAT fails, discard the disk since it is probably unusable.

Disk full—write not completed

Explanation: EDLIN. An End Edit command ended abnormally because the disk is full (not enough free space to save the entire file). Any editing done to the file is lost.

Action: Obtain a fresh diskette, copy the file on to the fresh diskette, and start editing again.

Disk not compatible

Explanation: FORMAT. You cannot use the DOS FORMAT command to format a diskette using the drive you specified. This message informs you that the drive you specified is not supported by the IBM device interfaces that FORMAT requires.

Action: Obtain a compatible disk drive.

Disk unsuitable for system disk

Explanation: FORMAT. A defective track was detected where the DOS files were to reside. The diskette can be used only for data.

Action: Use another disk if you wish to copy DOS files.

Diskettes compare OK

Explanation: DISKCOMP. Informational message. The two diskettes just compared contain identical information.

Action: No action required.

Diskette is not a backup diskette

Explanation: BACKUP and RESTORE. The diskette was not created by BACKUP. The first file on a backup diskette must be BACKUPID.@@@.

Action: Retry the command with the correct diskette.

Divide overflow

Explanation: DOS. A program tried to divide a number by zero, or a logic error caused an internal malfunction. The program ends and you return to DOS.

Action: Correct the programming error and continue. If this is a purchased program, take it back to your dealer.

Do you see the leftmost 9? (Y/N)

Explanation: MODE. ,R,T was specified.

Action: Respond Y or N. This prompt is repeated until you respond Y.

Do you see the rightmost 9? (Y/N)

Explanation: MODE. ,L,T was specified.

Action: Respond Y or N. This prompt is repeated until you respond Y.

Do you wish to use the entire fixed disk for DOS (Y/N)................? [ ]

Explanation: FDISK. When the "Create DOS Partition" option is used on the current fixed disk and the fixed disk has never been set up, this question is asked.

Action:

- If you enter **Y**, the entire current fixed disk is used for DOS and it will be made active.
- If you enter **N**, you are asked to enter the limits of the DOS partition you want to create.

DOS partition created

Explanation: FDISK. Informational message. A DOS partition has been created on the fixed disk.

Action: You will need to run the FORMAT command on the DOS partition before you can store files on the fixed disk.

DOS partition deleted

Explanation: FDISK. Informational message. The DOS partition no longer exists on the fixed disk.

Action: No action required.

Dup record too complex

Explanation: LINK. The problem is caused by too many structures or DUP statements in the object module created from an assembler source program.

Action: Reduce the number of structures or DUP statements in the assembler source program, create a new object module, and retry LINK.

Duplicate filename or file not found

Explanation: RENAME. You tried to rename a file to a filename that already exists on the disk, or the file to be renamed could not be found on the specified (or default) drive.

RENAME is warning you that you are using the same name for two files, or else it can't find the file you are trying to rename.

Action: Did you type the filename correctly? Take a second look at the filename you want to change, and re-enter the RENAME command.

E

Enter the number of the partition you want to make active...............: [ ]

Explanation: FDISK. The "Change Active Partition" option is requesting you to enter the number of the partition you want to make active.

Action: Type the number of the partition that you want to make active on the current fixed disk. Then press the Enter key.

> **Note:** The partitions are displayed above the prompt.

Enter partition size............: [dddd]

Explanation: FDISK. The "Create DOS Partition" option is requesting that you enter the size of the partition you wish to create.

Action: The number shown in the brackets is the default size. If you only press Enter, that size will be used as the partition size. Otherwise, enter the desired size, then press Enter.

Enter primary file name

Explanation: COMP. DOS asks you for primary filename.

Action: Enter the filespec of the first of two files to be compared.

Enter 2nd file name or drive id

Explanation: COMP. DOS asks you for filespec of second of the two files you want compared.

Action: Enter the filespec of the second of two files to be compared, or just enter the drive letter and/or path if the filename is the same as the primary filename.

Enter starting cylinder number..: [dddd]

Explanation: FDISK. The "Create DOS Partition" option is requesting that you enter the starting cylinder number for the DOS partition you are creating. The value in the brackets is the default value. It is the starting cylinder of the largest piece of free space on the current fixed disk.

Action: Type the starting cylinder number and press Enter, or just press Enter to use the default value.

Entry Error

Explanation: EDLIN. EDLIN has detected a syntax error.

Action: Correct the syntax error on the last command.

Entry has a bad attribute (or size or link)

Explanation: CHKDSK. This message may begin with one or two periods, indicating which entry in the subdirectory was in error. One period indicates the current directory is in error. Two periods mean the parent directory is in error. If you did *not* enter the /F parameter, no corrective action is taken.

Action: Enter: `CHKDSK /F`

CHKDSK will then try to correct the error. If you don't enter the /F parameter, CHKDSK does not write any corrections on the disk.

EOF mark not found

Explanation: COMP. COMP could not find the end of valid data in the last block of the files being compared. This message usually occurs when comparing nontext files; it should not occur when comparing text files.

Action: For more details, see the COMP command in Chapter 2.

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

Explanation: CHKDSK. Informational message. An error was found and you have not used the /F parameter.

CHKDSK will perform its analysis as though it were going to correct any errors detected, so that you can see the results of its analysis, but it will not actually write the corrections on the disk.

Action: No action required.

Error in EXE file

Explanation: DOS. An error was detected in the relocation information placed in the file by the LINK program. This may be due to a modification to the file.

Action:

- If you are using a purchased program, rerun the program using your backup copy.
- If you still have trouble see your authorized dealer.
- If you are using a program you wrote yourself, go through the LINK procedure again.

Error in EXE/HEX file

Explanation: DEBUG. The file contained invalid records or characters.

Action: Get another copy of the program, and run DEBUG again.

Error loading operating system

Explanation: Startup. A disk error occurred while attempting to load your operating system from fixed disk.

Action: Restart the system. If the error persists after several tries, restart the system, (you should start DOS from your DOS diskette) and use the SYS command to transfer a new copy of DOS to your fixed disk.

Error reading fixed disk.

Explanation: FDISK. The FDISK program was unable to read the startup record of the current fixed disk after five tries.

Action: Try the FDISK program again. If after several tries you still get the same error, consult the IBM *Guide to Operations* book, "Problem Determination" section. If you still can't solve the problem, see your IBM Personal Computer Dealer.

Error writing fixed disk.

Explanation: FDISK. The FDISK program was unable to write the startup record of the current fixed disk after five tries.

Action: Try the FDISK program again. If after several tries you still get the same error, consult the IBM *Guide to Operations* book, "Problem Determination" section. If you still can't solve the problem, see your IBM Personal Computer Dealer.

Error writing to device

Explanation: Commands. Informational message. DOS encountered an I/O error when writing output to a device. The device is unable to handle the number of bytes requested.

Action: Change amount of data in the file and retry the command.

Errors on list device indicate that it may be off-line. Please check.

Explanation: PRINT. The device being used for background printing is off line. This message only appears when the device is off line and you enter a new PRINT command.

Action: Make sure the printing device is connected and switched on.

EXE and HEX files cannot be written

Explanation: DEBUG. This error normally occurs when you load a .HEX or .EXE file, modify it, and then attempt to write the file back out to a diskette.

You should understand that .EXE and .HEX files contain loading information that is used to load the file. When DEBUG is executed, it loads the .EXE file while at the same time discarding the information. During direct execution of an .EXE file, the same thing happens. When you use DEBUG to write an .EXE file, the information is gone; therefore, a correct .EXE file cannot be generated. That is why you get this error message.

The error might also be caused by the fact that the data consists of a .COM file loaded in with DEBUG and you are now trying to write it to an .EXE or .HEX file. This is not possible. The data would require a backwards conversion that DEBUG doesn't support.

Action: If you need to look at the file's control information you can rename the file using a different extension, then execute DEBUG. DEBUG then *reads* the file in instead of loading it, and you can examine the control portion of the file.

EXEC failure

Explanation: Commands. DOS encountered an error while reading a command from disk, or the FILES= command in the configuration file (CONFIG.SYS) does not specify a large enough value.

Action: Increase the FILES= value. Restart DOS. If restarting DOS does not work, then there may be a problem with the disk itself.

F

File allocation table bad, drive x
Abort, Retry, Ignore?

Explanation: DOS.

Action: See the message **Disk error reading drive x** under "Device Error Messages" at the beginning of this appendix. If this error persists, the disk is unusable and should be formatted again.

File AND File

Explanation: COMP. Informational message. This message indicates the full path and filenames of the two files being compared.

Action: No action required.

File canceled by operator

Explanation: PRINT. Informational message. This message appears on the printer after you cancel the printing of a file to serve as a reminder that the printout is incomplete.

Action: No action required.

File cannot be copied onto itself

Explanation: COPY. You tried to COPY a file and place the copy (with the same name as the original) in the same directory and on the same disk as the original file.

Action: Change the name given to the copy, or put it in a different directory, or put it on another disk.

File creation error

Explanation: DOS and commands. An unsuccessful attempt was made to add a new filename to the directory or to replace a file that was already there.

Action: If the file was already there, check whether the file is marked read only and cannot be replaced. Otherwise, run CHKDSK to determine if the directory is full, or if some other condition caused the error.

File is cross-linked:
on cluster xx

Explanation: CHKDSK. This message appears twice for each cross-linked cluster number, naming the two files in error. The same data block is allocated to both files.

Action: No corrective action is taken automatically. You must correct the problem by doing the following:

1. Make copies of both files (use COPY command).
2. Delete the original files (use ERASE command).
3. Review the files for validity and edit as necessary.

File is currently being printed
File is in queue

Explanation: PRINT. Informational message. These messages appear together when you issue a PRINT command with no parameters. They occur individually when you queue the first or a subsequent file for printing.

Action: No action required.

File not found

Explanation: DOS and commands. A file named in a command or command parameter does not exist in the directory of the specified (or default) drive.

Action: Retry the command using the correct filename.

Files are different sizes

Explanation: COMP. Informational message. The sizes of the files to be compared do not match. This means that a comparison cannot be done because one of the files contains data which does not match the data in the other file.

Action: No action required.

Files compare OK

Explanation: COMP. Informational message. The two files just compared contain identical information.

Action: No action required.

First cluster number is invalid, entry truncated

Explanation: CHKDSK. Informational message. The file whose name precedes this message contains an invalid pointer to the data area. If you specify /F parameter the file is truncated to a zero length file.

Action: No action required.

Fixed disk already has a DOS partition.

Explanation: FDISK. You chose the "Create DOS Partition" option for a fixed disk that already has a DOS partition.

Action: Return to the Main Menu and choose option 4, **Display Partition Data.** Read Chapter 3 before continuing.

Fixup offset exceeds field width

Explanation: LINK. An assembler instruction refers to an address with a NEAR attribute instead of a FAR attribute.

Action: Edit assembler source program and process again.

Fixups needed – base segment (hex):

Explanation: EXE2BIN. The source (.EXE) file contains information indicating that a load segment is required for the file.

Action: Specify the absolute segment address at which the finished module is to be loaded.

> Note: We do not recommend using such a program as a .COM file because the program is dependent upon being loaded at a specific memory location.

FOR cannot be nested

Explanation: Batch. More than one FOR subcommand was found on one command line in the batch file.

Action: Use only one FOR subcommand per command line. Then retry command.

Format failure

Explanation: FORMAT. A disk error was encountered while creating the target disk.

Action: Try format again–if the error persists, then the disk is unusable.

Formatting while copying

Explanation: DISKCOPY. Informational message. The target diskette contains unformatted tracks. DISKCOPY will format the remainder of the target diskette as it copies data.

Action: No action required.

Note: If this message is followed by the message **Incompatible drive types**, you tried to copy a dual-sided diskette to a drive that does not have dual-sided capability. This cannot be done. Processing ends and the target diskette contains no useful data.

I

Illegal Device Name

Explanation: MODE. The specified printer must be:

- LPT1:
- LPT2:

  or,
- LPT3:

The specified Asynchronous Communications Adapter must exist and be:

- COM1:

  or,
- COM2:

There must be no more than one blank between MODE and its parameters.

Action: Use the correct device name and retry the command.

Incompatible diskette or drive types

Explanation: DISKCOMP. The first diskette was successfully read on both sides, but the second diskette could only be read on the first side.

Either the second drive or diskette is single sided or the first diskette contains 9-sectors per track and the second diskette contains only 8-sectors per track.

Action: The diskette or drive types must be compatible. Make sure you put the correct diskette in the drive. Retry command.

Incompatible drive types

Explanation: DISKCOPY. The source diskette and drive are dual-sided, but the target drive has only single-sided capability. The target diskette contains no useful data.

Action: Use compatible drive types and retry command.

Incompatible system size

Explanation: SYS. The target diskette contained a copy of DOS that is smaller than the one being copied. The system transfer does not take place.

Action: Format a blank diskette (use the FORMAT /S command) and then copy any files to the new diskette.

Incorrect DOS version

Explanation: Commands. The command you just entered requires a different version of DOS than the one you are using.

Action: Obtain correct version of DOS and retry command.

Insert backup diskette *xx*
in drive *x*:
Strike any key when ready

Explanation: RESTORE.

Action: Insert the backup diskette(s) in sequence in accordance with the prompt. Press any key and RESTORE will continue.

Insert backup diskette *xx*
in drive *x*:
Warning! Diskette files will be erased
Strike any key when ready

Explanation: BACKUP.

Action: Insert the next diskette to be used for the backup. Use DOS formatted diskettes only. Press any key and BACKUP continues.

> Note: Any files on the backup diskette are erased.

Insert COMMAND.COM disk in drive *x*:
and strike any key when ready

Explanation: DOS. DOS is attempting to reload the command processor, but COMMAND.COM is not on the drive that DOS was started from.

Action: Insert the DOS diskette in the indicated drive and press any key.

Press any key to begin recovery of the file(s) on drive X:

Explanation: RECOVER.

Action: Insert the diskette to be recovered in the indicated drive and press any character key.

Insert disk with batch file and strike any key when ready

Explanation: DOS. The diskette that contained the batch file being processed was removed. The batch processor is trying to find the next command in the file.

Action: Insert the diskette in the appropriate drive and press any key. Processing will then continue.

Insert DOS disk in X: and strike any key when ready

Explanation: SYS and FORMAT. FORMAT or SYS is trying to load the DOS files, but the indicated drive X: does not contain the DOS diskette.

Action: Follow the prompt and insert the DOS diskette. Press any key and processing continues.

Insert DOS diskette in drive A:
Press any key when ready . . .

Explanation: FDISK. You have successfully created the DOS partition on the current fixed disk.

Action: Insert the DOS diskette into drive A and press any key. This restarts your IBM Personal Computer.

The current fixed disk is now assigned a fixed disk letter and you can now FORMAT the fixed disk.

Insert first diskette in drive x
Insert second diskette in drive x

Explanation: DISKCOMP.

Action: Insert the first (or second) of the two diskettes to be compared into the indicated drive. One or both of these messages is followed by the message **Strike any key when ready.** Press a key and the comparison starts.

Insert source diskette in drive x
Insert target diskette in drive x

Explanation: DISKCOPY.

Action: Insert the appropriate diskette into the indicated drive, and press any key when asked. The copying process starts.

Insufficient disk space

Explanation: DOS and commands. The disk does not contain enough free space to contain the file being written.

Action: If you suspect this condition is invalid, run CHKDSK to determine the status of the disk. Otherwise, use another disk and retry the command.

Insufficient memory

Explanation: Commands. The amount of available memory is too small to allow these commands to function.

Action: Change the BUFFERS= parameter in the CONFIG.SYS file to a smaller value (if you have specified BUFFERS=), restart the system, and try the command again. If the message still appears, your system does not have enough memory to execute the command.

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

Explanation: CHKDSK. You instructed CHKDSK to create files from the "lost" data blocks it has found, but the root directory is full, and all of the lost chains could not be recovered into files.

Action:

1. Copy some of the recovered files to another disk for further examination.

2. Delete the recovered files from the disk you are checking.

3. Run CHKDSK again to recover the remainder of the lost data.

Insufficient space on disk

Explanation: DEBUG. A write command was issued to a disk that doesn't have enough free space to hold the data being written.

Action: If you are writing to a diskette, you can insert a diskette that has enough free space, then reissue the write command. Otherwise, you should erase files from the disk and run DEBUG again.

Intermediate file error during pipe

Explanation: DOS. DOS is unable to create one or both of its intermediate files because the default drive's root directory is full, or DOS is unable to locate the piping files, or the disk does not have enough space to hold the data being piped.

Action: Erase some files from the default drive's root directory, and reissue the command that failed. If you get the same message, one of the programs in the command line has erased one or both of the piping files. Correct the program and reissue the command line.

Invalid baud rate specified

Explanation: MODE.

Action: Specify the baud rate as 110, 150, 300, 600, 1200, 2400, 4800, or 9600 (you need specify only the first two characters of the number).

Invalid characters in volume label

Explanation: FORMAT. One or more of the characters you entered in the volume label is not a valid filename character, or the name contained a period (volume labels contain 1 to 11 valid characters without a period).

Action: Retry using valid characters.

Invalid COMMAND.COM in drive n

Explanation: DOS. When DOS tried to reload the command processor, the copy of COMMAND.COM on the disk was found to be an incorrect version.

Action: Insert the correct DOS diskette and press any key to continue.

Invalid date

Explanation: DATE. You entered an invalid date or delimiter. The only valid delimiters in a date entry are hyphens (–) and slashes (/).

Action: Re-enter valid date.

Invalid device

Explanation: CTTY. The device name you specified is an invalid name to DOS.

Action: Retry command using valid device name.

Invalid directory

Explanation: DOS and commands. One of the directories in the specified path does not exist.

Action: Retry command using valid directory.

Invalid drive in search path

Explanation: DOS. An invalid drive specifier was found in one of the paths specified in the PATH command.

This message appears when DOS attempts to locate a command or batch file, not at the time you issued the erroneous PATH command.

Action:

1. Enter PATH. This displays the *PATH's* you previously defined.

2. Find the invalid specifier.

3. Re-enter the PATH command with the valid drive specifier and the paths you desire.

Invalid drive specification

Explanation: DOS and commands. An invalid drive specification was just entered in a command or in one of its parameters.

Action: Re-enter the command using a valid drive specifier.

Invalid format file

Explanation: LINK. A library is in error.

Action: Restore library file from your backup disk and try again.

Invalid number of parameters

Explanation: Commands. You have specified too few or too many parameters for the command you issued.

Action: Review this command in the "Commands" section of this book.

Invalid numeric parameter

Explanation: LINK. The numeric value is not in digits.

Action: Run LINK again and name the numeric parameter with values of 0 through 9 for each digit.

Invalid object module

Explanation: LINK. Object module(s) were incorrectly formed or unobserved errors occurred during compilation. The disk may be bad.

Action: Recompile the module to either the same or a different disk.

Invalid parameter

Explanation: DOS and commands. One or more of the parameters entered for these commands is not valid.

Action: If the program expects a drive specifier, enter a colon following the drive letter. In other cases, make sure the character following the slash (/) is valid for the program being run.

Invalid parameters

Explanation: MODE.

- No parameters were entered
- The first parameter character was other than L, or C
- The first parameter was other than **40, 80, BW40, BW80, CO40, CO80, MONO, L, R**
- The display adapter the parameter refers to is not present in the machine.

Action: Check the preceding list and correct accordingly.

Invalid partition table

Explanation: Startup. While attempting to start DOS from your fixed disk, the start-up procedures detected invalid information in the disk's partition information.

Action:

1. Start DOS from the diskette.
2. Use the FDISK command to examine and correct the fixed disk partition information.

Invalid path

Explanation: TREE. Tree was unable to use a directory whose name was found in another directory.

Action: Run CHKDSK to find out what is wrong with the directory structure.

Invalid path, not directory or directory not empty

Explanation: RMDIR.

- The specified directory was not removed because one of the names you specified in the path was not a valid directory name

  or,

- The directory you specified still contains entries for files or other subdirectories (with the exception of the . and .. entries)

- You cannot remove a current directory

Action: Try one of the following:

- Correct the invalid directory name in the path

- Delete any files or remove any subdirectories in the directory

Invalid path or file name

Explanation: COPY. You specified a directory or file name that does not exist.

Action: Use correct name. Check for the following, then retry command:

- Correct spelling of names

- Valid directory names

- Existence of file in the subdirectory specified

Invalid subdirectory

Explanation: CHKDSK. Invalid information was detected in the subdirectory whose name precedes this message.

Action: CHKDSK will attempt to correct the error if you have used the /F parameter. For more specific information about the nature of the error, run CHKDSK with the /V parameter.

Invalid switch

Explanation: LINK. The character indicated on the preceding line is not a valid linker parameter (switch).

Action: Review Chapter 7, “The Linker (Link) Program” to determine which characters are valid link parameters (switches).

Invalid time

Explanation: TIME. An invalid time or delimiter was entered.

Action: Re-enter the correct time. The only valid delimiters are:

- A colon (:) between the hours and minutes
- A colon (:) between the minutes and seconds
- A period (.) between the seconds and hundredths of a second.

L

Label not found

Explanation: Batch. Informational message. A GOTO command named a label that does not exist in the batch file. This caused the system to read to the end of the batch file, ending batch processing.

Action: If you do not want the GOTO to exit the batch file, edit the batch file and put the label in the desired location.

Line too long

Explanation: EDLIN. Upon replacing a string, the replacement caused the line to expand beyond the 253-character limit. The REPLACE text command is ended abnormally.

Action: Split the long line into shorter lines; then issue the REPLACE text command again.

List output is not assigned to a device

Explanation: PRINT. The device you named as the PRINT list device is not recognized as a valid device.

Action: Reissue the PRINT command and reply with a valid list device name when asked.

LPT#: not redirected.

Explanation: MODE. Informational message. The parallel printer will now receive its own output, even if this printer's output had previously been redirected to a serial device.

This message is provided for your information and indicates cancellation of any previous redirection which may have been in effect, because you have set the printer width or vertical spacing.

Action: No action required.

LPT#: redirected to COM*n*:

Explanation: MODE. Informational message.

This message is provided for your information and indicates that any request that would normally have gone to the parallel printer LPT# (#=1, 2, or 3) is sent instead to the serial device COM*n* (*n*=1 or 2).

Action: No action required.

LPT#: set for 80

Explanation: MODE. Informational message. You tried to set the printer line length to 80 characters by requesting standard type format.

If the attempt is unsuccessful, an error message follows this message on the screen.

Action: No action required.

LPT#: set for 132

Explanation: MODE. Informational message. You tried to set the printer line length to 132 characters by requesting compressed type format.

If the attempt is unsuccessful, an error message will follow this message on the screen.

Action: No action required.

M

Maximum available space is *xxxx* cylinders at cylinder *xxxx*.

Explanation: FDISK. Informational message. The "Create DOS Partition" option displays the largest available piece of space on the current fixed disk. These numbers are also used as the defaults for the two prompts that will follow.

Action: No action required.

Memory allocation error
Cannot load COMMAND, system halted

Explanation: DOS. A program destroyed the area in which DOS keeps track of available memory.

Action: Restart DOS.

Missing operating system

Explanation: Startup. When you tried to start DOS from fixed disk, the startup procedures determined that the DOS partition was marked "bootable" (startable), but that it doesn't contain a copy of DOS.

Action: Start DOS from diskette and use FORMAT with the /S parameter to place a copy of DOS on the fixed disk. You might want to back up your files before doing the FORMAT.

Must specify destination line number

Explanation: EDLIN. A Move or Copy command was entered without a destination line number.

Action: Re-enter the command with a valid destination line number.

N

Name of list device [PRN]:

Explanation: PRINT. This message appears the first time you start print after DOS has been restarted.

Action: Reply with the reserved device name which is to receive the printed output, or simply press Enter if the first parallel printer [PRN] is to be used.

No DOS partition to delete.

Explanation: FDISK. You chose the "Delete DOS Partition" option but there was no DOS partition on the current fixed disk.

Action:

1. Return to the Main Menu.

2. Select the **Display Partition Data** to review.

3. Proceed with your next choice.

No free file handles
Cannot start COMMAND, exiting

Explanation: DOS. An attempt to load a second copy of the command processor failed because there are too many files open.

Action: Increase the number in the FILES= command in the configuration file (CONFIG.SYS), and restart DOS.

No fixed disks present

Explanation: FDISK. The FDISK program was run on an IBM Personal Computer that:

- does not have a fixed disk, or

- has a fixed disk in the expansion unit and the expansion unit is not powered on, or

- has a fixed disk that is not properly installed

Action: From the above list find out what caused the problem and take appropriate action. Check to make sure the expansion unit is powered ON first.

No object modules specified

Explanation: LINK. You did not name any object modules in the command line or in response to the prompt.

Action: Name the object modules, since the linker needs some files to link.

No partitions to make active.

Explanation: FDISK. You chose the "Change Active Partition" option but there were no partitions on the current fixed disk to be made active.

Action: Use the "Create DOS Partition" option to create a partition, then the "Change Active Partition" option to make it the active partition.

No path

Explanation: PATH. Informational message. There is currently no alternate path for DOS to search to find commands and batch files if it does not find them in the specified (or current directory).

Action: Informational message unless you want to define a set of paths. If so, enter PATH and the set of paths you want. Then press Enter.

No room for system on destination disk

Explanation: SYS. The destination diskette does not contain the required reserved space for DOS; therefore, the system cannot be transferred.

Action: Format a blank diskette (use the FORMAT /S command), then copy any other files to the new diskette.

No room in directory for file

Explanation: EDLIN. The directory on the specified disk is full. Your editing changes are lost.

Action: Make sure that your disk has available directory entries and run EDLIN again.

No space for a xxxx cylinder partition.

Explanation: FDISK. You entered a "Partition Cylinder Size" that is larger than the largest piece of free space on the disk.

Action: Enter a smaller number.

No space for a xxxx cylinder partition at cylinder xxxx.

Explanation: FDISK. You requested a partition to be created at a place on the current fixed disk and there is no space at that place to create a DOS partition.

Action: Look for typographical errors. Reinvestigate your partitioning requirements.

No space to create a DOS partition.

Explanation: FDISK. You chose the "Create DOS Partition" option on the current fixed disk which has no space to create a DOS partition.

Action: Remove or reduce the size of the existing partition. Then run FDISK again to create the DOS partition(s).

No subdirectories exist

Explanation: TREE. Informational message. The specified drive contains only a root directory. Therefore, there is no directory path to display.

Action: No action required.

Non-System disk or disk error
Replace and strike any key when ready

Explanation: Startup. No entry exists for IBMBIO.COM or IBMDOS.COM in the directory; or a disk read error occurred when you started up the system.

Action: Insert a DOS diskette in drive A and then restart your system.

Not enough room to merge the entire file

Explanation: EDLIN. Informational message. A Transfer command was unable to merge the entire contents of the specified file because of insufficient memory. Only part of the file was merged.

Action: Use the Write lines command to write some of the lines out to disk and retry the Transfer command. If this does not work either reduce the size of one of the files being merged, or install more memory.

Not found

Explanation: EDLIN. Informational message. EDLIN count not find the string specified by the REPLACE text or SEARCH text commands within the specified range of lines. Or, if a search is resumed by replying **N** to the **OK?** prompt, no further occurrences of the string were found.

Action: Check to be sure you properly used upper and lower case for the string to be searched.

O

Out of environment space

Explanation: DOS. Informational message. DOS was unable to accept the SET command you just issued because it was unable to expand the area in which the environment information is kept.

This normally occurs when you try to add to the environment after loading a program which makes itself resident (PRINT, MODE, or GRAPHICS for example).

Action: No action required.

Out of space on list file

Explanation: LINK. There is not enough disk space for the List file.

Action: Use a disk with enough free space to hold the file.

Out of space on run file

Explanation: LINK. There is not enough disk space for the Run file (.EXE).

Action: Use a disk with enough free space to hold the file.

Out of space on VM.TMP

Explanation: LINK. No more disk space remained to expand the VM.TMP file.

Action: Use a disk with enough free space to hold this file.

P

Parameters not compatible

Explanation: FORMAT. You attempted to use two parameters that are not compatible with each other (/B and /V for example).

Action: Review the FORMAT command. Correct parameter and re-enter the command.

Parameter not compatible with fixed disk

Explanation: FORMAT. You wrongly specified the /1 or /8 parameter while formatting a fixed disk. Neither of these parameters is valid for a fixed disk.

Action: Review the FORMAT command. Correct parameter and re-enter the command.

Partition 1 is already active

Explanation: FDISK. Informational message. Partition 1 is the only partition defined and it is already marked as active.

Action: No action required.

Partition xx made active

Explanation: FDISK. Informational message. Partition xx is now marked as bootable.

Action: No action required.

Press any key to begin formatting x:

Explanation: FORMAT. The fixed disk (drive x) is about to be formatted. Formatting will lose track of all previously existing data on the disk.

Action: If you do *not* want the disk formatted, press Ctrl-Break. If you *do* want the disk formatted, press a character key.

Print queue is empty

Explanation: PRINT. Informational message. There are currently no files being processed by PRINT.

Action: No action required.

Print queue is full

Explanation: PRINT. You tried to add more than the limit of ten files to the print queue.

Action: Wait until a file is printed before you add another file to the print queue.

Printer error

Explanation: MODE. The MODE command (option 1) was unable to set the printer mode because:

- There is an I/O error
- The printer is out of paper (or POWER OFF)
- There is a printer time out (not ready) condition
- The printer is offline

Action: Determine which of above conditions caused the error message and take corrective action.

Printer lines per inch set

Explanation: MODE. Informational message. You tried to set the printer vertical spacing to the specified 6 or 8 lines per inch.

Action: If the attempt was unsuccessful, an error message will follow this message on the screen.

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

Explanation: CHKDSK. The file allocation table identification byte contains invalid information. Either the disk was not formatted by DOS or has it become badly damaged.

Action: If you did not use the /F parameter, and you reply Y, CHKDSK will indicate its possible corrective actions without actually changing the disk. We recommend doing this first, before you consider using the /F switch and replying Y.

Processing cannot continue

Explanation: CHKDSK. Informational message. This message is followed by another message which explains why CHKDSK cannot continue.

This message is normally issued when there is not enough memory.

Action: No action required.

Program size exceeds capacity of LINK

Explanation: LINK. Load module is too large for processing.

Action: Reduce the size of your program.

Program too big to fit in memory

Explanation: DOS. The file containing the external command cannot be loaded because it is larger than the available free memory.

Action: Reduce the number in the BUFFERS= parameter in your CONFIG.SYS file (if you have specified BUFFERS=), restart your system, and reissue the command.

If the message reappears, your system does not have enough memory to execute the command.

R

Reinsert new diskette for drive X

Explanation: FORMAT. This message, which usually occurs after you enter FORMAT /S, means:

- DOS filled the memory with system files, but could not read all of the files into memory because of insufficient memory size.
- After asking for the new diskette, FORMAT started formatting it and wrote all of the files in memory on the new diskette.
- FORMAT then asked that the DOS diskette be inserted so it could finish loading the rest of the DOS files into memory.

Action: FORMAT is now asking you to insert the new diskette again so it can finish the task of writing the DOS files onto the new diskette.

Requested stack size exceeds 64K

Explanation: LINK.

Action: Specify a stack size of less than or equal to 64K bytes when the STACK SIZE: prompt appears.

Resident part of PRINT installed

Explanation: PRINT. Informational message. The message appears the first time you use the PRINT command.

This message indicates that a program has been loaded into memory to handle subsequent PRINT commands. Available memory for your applications has been reduced by approximately 3200 bytes.

Action: No action required.

Resident portion of MODE loaded

Explanation: MODE. Informational message.

This message indicates that when MODE is invoked for a non-screen-setting function it is sometimes necessary to load a portion of code to be made permanently resident.

Action: No action required.

S

Sector size too large in file <filename>

Explanation: Startup. The device driver named in <filename> specifies a device sector size larger than the devices previously defined to DOS.

Action: Reduce sector size to conform with the sector size of DOS.

If this is a purchased program, return it to your dealer.

Segment size exceeds 64K

Explanation: LINK. This message indicates that you attempted to combine identically named segments which resulted in a segment requirement of greater than 64K bytes. You cannot address more than 64K bytes.

Action: Change segment names in object modules and try link again.

Specified command search directory bad

Explanation: Commands. Invalid path name specified.

Action: Enter a correct SET COMSPEC= command, or correct the problem that caused the command processor's directory path to become invalid.

Stack size exceeds 65535 bytes

Explanation: LINK. Informational message. The size specified for the stack must be less than or equal to 65535.

Action: No action required.

Symbol defined more than once

Explanation: LINK. The Linker found two or more modules that define a single symbol name

Action: Check for the following:

- Are you sure you're not linking the same files twice?
- Does one of the modules being linked have a symbol incorrectly identified as *public* instead of *external*?

Symbol table capacity exceeded

Explanation: LINK. Many, very long names were entered. The names exceeded approximately 50K bytes.

Action: Use shorter and/or fewer names.

Syntax error

Explanation: DOS. The command you entered is improperly formatted.

Action: Check to make sure you have used the correct format for this command.

System will now reboot
Insert DOS diskette in drive A:
Press any key when ready

Explanation: FDISK. Informational message. FDISK requires the system to restart in order to recognize the fixed disk (so you can format it.)

Action: Place DOS diskette in Drive A: and press any key.

T

Target diskette may be unusable

Explanation: DISKCOPY. This message follows an unrecoverable read, write, or verify error message. The copy on the target diskette may be incomplete because of the unrecoverable I/O error.

Action:

- If error is on the target diskette, get a fresh diskette for your target, and retry the DISKCOPY command.
- If the error is on the source diskette, copy all files from the source diskette to another diskette. Then try to reformat the source diskette.

Target diskette write protected
Correct, then strike any key

Explanation: DISKCOPY. You are trying to produce a copy on a diskette that is write protected.

Action: Either remove the write protect tab, or use another diskette that is not write protected.

Terminate batch job (Y/N)?

Explanation: DOS. This message appears when you press Ctrl-Break while DOS is processing a batch file.

Action: Press Y to stop processing the batch file. Pressing N only ends the command that was executing when Ctrl-Break was pressed; processing resumes with the next command in the batch file.

The current active partition is *x*.

Explanation: FDISK. Informational message. The "Change Active Partition" option displays the active partition on the current fixed disk.

Action: No action required.

The last file was not restored

Explanation: RESTORE. You stopped RESTORE before it completely restored the last file listed, or there was not enough room on the fixed disk and RESTORE deleted the partially restored file.

Action: If RESTORE has ended, you can re-enter the RESTORE command with the filename of the file(s) not restored to continue from the point where RESTORE stopped.

If the problem occurred because you ran out of room on the fixed disk, you must evaluate which files to keep and which ones to delete. Then continue the execution of RESTORE.

There was/were *number* errors detected

Explanation: LINK. This message is displayed for your information at the end of the link session.

Action: No action required.

Too many external symbols in one module

Explanation: LINK. The limit is 256 external symbols per module. This message indicates that the limit was exceeded.

Action: Break up some modules.

Too many groups

Explanation: LINK. The limit is 10 including DGROUP.

This message indicates that the limit was exceeded.

Action: Reduce the number of groups.

Too many libraries specified

Explanation: LINK. The limit is 8 libraries.

This message indicates that the limit was exceeded.

Action: Reduce the number of libraries.

Too many overlays

Explanation: LINK. The limit is 64 overlays.

This message indicates that the limit was exceeded.

Action: Reduce the number of overlays.

Too many public symbols

Explanation: LINK. The limit is 1024 public symbols.

This message indicates that the limit was exceeded.

Action: Reduce the number of public symbols.

Too many segments or classes

Explanation: LINK. The limit is 247 segments or classes taken together.

This message indicates that the limit was exceeded.

Action: Reduce the number of segments or classes.

Total disk space is *xxxx* cylinders.

Explanation: FDISK. Informational message. This message shows the total space on the current fixed disk.

Action: No action required.

Track 0 bad—disk unusable

Explanation: FORMAT. Track 0 is where the boot record, file allocation table, and directory must reside. The disk is unusable.

Action: Obtain another disk and retry the FORMAT command.

Tree past this point not processed

Explanation: CHKDSK. Informational message. CHKDSK is unable to continue processing past the directory path currently being examined. The cause of the error is displayed in the previous message.

Action: No action required.

U

Unable to create directory

Explanation: MKDIR. The directory you want to create already exists, or one of the directory path names you specified could not be found, or you attempted to add a directory to the root directory and it is full. Make sure a file by that name does not already exist in that directory.

Action: Do the following:

- Check to see if a directory by that name exists in the parent directory (or current directory)
- Recheck all your directory names to make sure they are valid
- Use CHKDSK to see if your directory is full

Unable to write BOOT

Explanation: FORMAT. The first track of the diskette or DOS partition is bad. The BOOT record could not be written on it. The diskette or DOS partition is not usable.

Action: Obtain another diskette and retry the FORMAT command.

Unexpected end-of-file on library

Explanation: LINK. Informational message. This is caused by an error in the library file. This usually means that the object file contains bytes that are of the same value as end-of-file. If this occurs, LINK continues processing to the physical end-of-file as given in the directory.

Action: No action required.

Unexpected end of file on VM.TMP

Explanation: LINK. Informational message. The diskette containing VM.TMP was removed.

Action: No action required.

Unrecognized command in CONFIG.SYS

Explanation: Startup. An invalid command was detected in the configuration file CONFIG.SYS.

Action: Edit the file, correct the invalid command, and restart DOS.

Unrecognized switch

Explanation: LINK. The character(s) specified by *z* do not uniquely identify a linker parameter (switch).

Action: See the message, "Invalid Switch."

Unrecoverable format error on target
Target diskette unusable

Explanation: DISKCOPY. An unrecoverable error was encountered while formatting the target diskette. The diskette contains no usable data.

Action: Obtain another diskette and retry the DISKCOPY command.

Unrecoverable read error on drive *x*
Track *xx*, side *x*

Explanation: DISKCOMP. Four attempts were made to read the data from the diskette in the specified drive. The data could not be read from the indicated track and side.

Action: If the error occurred on the target diskette (just created by DISKCOPY) get a fresh diskette and retry the DISKCOPY and DISKCOMP commands. Otherwise, use:

```
COPY *.*
```

to copy all files from the damaged diskette to another diskette. Then reformat the bad diskette or else discard it.

Unrecoverable read error on source
Track *xx*, side *x*

Explanation: DISKCOPY. Four attempts were made to read the data from the source diskette. DISKCOPY continues copying, but the copy may contain incomplete data.

Action: Use:

```
COPY *.*
```

to copy all files from the damaged diskette to another diskette. Reformat the bad diskette, or else discard it.

Unrecoverable verify error on target
Track *xx*, side *x*

Explanation: DISKCOPY. Four attempts were made to verify the write operation to the target diskette. DISKCOPY continues copying, but the copy may contain incomplete data.

Action: Repeat the DISKCOPY command, or else use a different diskette.

Unrecoverable write error on target
Track *xx*, side *x*

Explanation: DISKCOPY. Four attempts were made to write the data to the target diskette. DISKCOPY continues copying, but the copy may contain incomplete data.

Action: Obtain a fresh diskette and re-enter the DISKCOPY command. Use FORMAT on the bad diskette to see if it can be reused. If it is a bad diskette, discard it.

Unresolved externals: list

Explanation: LINK. The external symbols listed were not defined in the modules or library files that you specified.

Action: Do not attempt to run the executable file created by the linker.

Make sure you specified all appropriate object modules and libraries. Check the source code for the program that caused the message and make corrections to that program.

V

VERIFY is OFF/ON

Explanation: VERIFY. Informational message.

Action: If *VERIFY is ON* is displayed, and *VERIFY is OFF* is what you want, enter:

```
VERIFY OFF
```

VM.TMP is an illegal file name and has been ignored

Explanation: LINK.VM.TMP cannot be used for an object filename. Informational message that is meant as a warning.

Action: No action required.

Volume label (11 characters, ENTER for none) ?

Explanation: FORMAT. You are requested to enter a 1 to 11 character volume label which will be written on the disk being formatted.

Action: If you do not want a volume label on the disk, press only the ENTER key.

W

Warning! Data in the DOS partition could be lost. Do you wish to continue....................? [ ]

Explanation: FDISK. The "Delete DOS Partition" option is warning you that if you continue, all data in the DOS partition on the current fixed disk could be destroyed.

Action: If you press Enter, the DOS partition will NOT be destroyed. If you *do* wish to delete the DOS partition, type Y and press Enter.

Warning-directory full

Explanation: RECOVER. There is insufficient directory space to recover more files.

Action: Copy some of the files to another disk, erase them from this disk, and run RECOVER again.

Warning! Diskette is out of sequence
Replace the diskette or continue
Strike any key when ready

Explanation: RESTORE. The backup diskette is not the next one in sequence.

Action: Replace the diskette unless you are sure no files on the diskette(s) you skipped would be restored. RESTORE will continue when you press a key. This message will be repeated if you try to skip a diskette which contains part of a file being restored.

Warning! File *xx*
is a read-only file
Replace the file (Y/N)?

Explanation: RESTORE. The indicated file is read only.

Action: Enter **Y** if you want to replace it or **N** if you do not. RESTORE will continue after you press ENTER. You will see this message only if you specified the /**P** option.

Warning! File *xx*
was changed after it was backed up
Replace the file (Y/N)?

Explanation: RESTORE. The indicated file on the fixed disk has a later date and time than the corresponding file on the backup diskette.

Action: Enter **Y** if you want to replace it with the backed up version or **N** if you do not. RESTORE will continue after you press ENTER. You will see this message only if you specified the /**P** option.

Warning! No files were found to back up

Explanation: BACKUP. No fixed disk files were found that matched the backup file specification.

Action: Make sure the criteria you specified for BACKUP is what you want. If so, this is an informational message.

Warning! No files were found to restore

Explanation: RESTORE. No backup diskette files were found that matched the restore file specification.

Action: Make sure the criteria you specified for BACKUP is what you want. Otherwise, this is an informational message.

Warning: no stack segment

Explanation: LINK. Informational message. None of the object modules specified contain a statement allocating stack space.

Action: No action required.

WARNING—Read error on EXE file

Explanation: EXE2BIN. An error occurred while reading the input file. EXE2BIN will attempt to continue, but the resulting file may be unusable.

Action:

1. Use the COPY *.* command to copy the diskette to a new diskette.
2. Copy your backup version of the .EXE file to this diskette.
3. Format the diskette that had the error to mark any bad tracks.
4. Discard the bad diskette, if necessary.

X

x is not a choice. Enter a choice.

Explanation: FDISK. You entered *x* which is not a choice for this question.

Action: Enter a valid choice.

x is not choice. Enter Y or N.

Explanation: FDISK. You entered *x* which is not a choice for this question.

Action: Enter Y or N.

XXXXXXXXXX bytes disk space freed

Explanation: CHKDSK. Informational message. Disk space marked as allocated was not associated with a file. If you used the /F parameter, the space was freed and made available.

Action: No action required.

xxxx error on file yyyy

Explanation: PRINT. This message appears on the printer. While attempting to read data from file yyyy for printing, a disk error of type xxxx was encountered. Printing of that file is stopped.

Action: Check that disk drive is ready.

xxx lost clusters found in yyy chains.
Convert lost chains to files (Y/N)?

Explanation: CHKDSK. CHKDSK located *xxx* blocks of the data area which were marked as allocated, but were not associated with a file. These clusters are assumed to contain "lost" data, and CHKDSK will ask whether you wish to free them, or to recover each chain into a separate file.

Action: If you reply **Y** and you have used the /F parameter, CHKDSK will recover each chain into a separate file, otherwise, if you reply **N**, CHKDSK frees the blocks up so they can be allocated to new files. If CHKDSK was specified (no /F), then messages displayed afterwards are informational (no corrective action was taken).

***** Backing up files to diskette xx *****

Explanation: BACKUP. Informational message. This message is followed by a list of files that were backed up on the indicated diskette.

Action: No action required.

***** Files were backed up xx/xx/xxxx *****

Explanation: RESTORE. Informational message. The files on the backup diskette were backed up on the indicated date.

Action: No action required.

***** Restoring files from diskette xx *****

Explanation: RESTORE. Informational message. This message is followed by a list of files that were restored from the indicated diskette.

Action: No action required.

——More——

Explanation: MORE. The screen is full and there is more data waiting to be displayed.

Action: Press any character to see the next full screen.

Numbers

10 Mismatches — ending compare

Explanation: COMP. Informational message. Ten mismatched locations were detected in the files being compared. COMP assumes that the files are so different that further comparisons would serve no purpose.

Action: No action required.

Appendix B. Running Compilers and Assemblers

Using Compilers and Assemblers With Fixed Disk

The following is a summary of how to run the IBM Personal Computer compilers and the IBM Personal Computer Macro Assembler with the Disk Operating System (DOS) Version 2.10. For the purposes of this appendix, the word *compile* will also refer to assemble.

1. Make sure to back up your original language diskettes using the techniques described within this book and/or the respective language books.

2. Make sure all source code you will compile is in the current directory of your disk.

3. Most of the language processors may be located in any directory at compile time (see exception list at the end of this appendix).

4. When Compiling you should use the command lines as described in your language book. You can modify the drive specifier in the command line to indicate the location of the particular language processor.

5. When Linking, use the IBM Personal Computer Linker, Version 2.10, provided with DOS Version 2.10. The instructions within this book and your language book will provide the necessary information for linking.

6. When running a program (.BAT, .COM, or a .EXE file) it is not always necessary for that program to be in the current directory (see the PATH command in Chapter 2 of this book). However, if the program requires another file at runtime; such as a data file or a Common Runtime Module, then those files must be in the current directory at runtime.

7. It is possible for COMMAND.COM to be overwritten in memory. Therefore, it is usually helpful to have a copy of COMMAND.COM in the root directory of the drive from which DOS was started.

Exceptions

- All IBM Personal Computer Language Products:
  - All files accessed by your program must be in the current directory at runtime.
  - Any Common Runtime Module used by your program must be in the current directory at runtime.
- IBM Personal Computer BASIC Compiler:
  - BASRUN.EXE, if used at runtime, must be in the current directory.

- IBM Personal Computer COBOL Compiler:
  - COBOL.COM and its overlays must be in the current directory at compile time.
  - To compile, you must use the command line and indicate the drive location of the overlays with the /C parameter.
  - COBRUN.EXE must be in the current directory at runtime.
- IBM Personal Computer Pascal Compiler:
  - Pascal requires a hex update (see "Pascal Hex Patch" in Appendix C) in order to run this compiler from a drive other than drive A.
  - PASKEY must be in the current directory at compile time.

Appendix C. Running the Pascal Compiler

Using Pascal Hex Patch With Fixed Disk

The IBM Personal Computer Pascal Version 1.00 Compiler requires that a special file called PASKEY be in the current directory on diskette drive A. The following hex update can be used so that the Pascal compiler will look on the default drive for this PASKEY file.

The following describes how to update your PAS1 diskette. Follow the instructions carefully. If you make a mistake don't panic. Just start again.

To update your PAS1 diskette, you need a blank formatted diskette. Put your DOS diskette in diskette drive A and the blank formatted diskette in diskette drive B. You must use the DOS DISKCOPY command to make an exact copy of your PAS1 diskette.

After the DISKCOPY program is started and you see the message Strike any key when ready, insert the ORIGINAL copy of the Pascal PAS1 diskette in diskette drive A. It must be the diskette from your IBM Personal Computer Pascal Compiler package or an exact copy of it. Anything else will not work. After you insert the diskettes, press the Spacebar to start the disk copy:

```
A>DISKCOPY A: B:

Insert source diskette in drive A:

Insert target diskette in drive B:

Strike any key when ready


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

Place your DOS diskette back in diskette drive B. You now have an exact copy of your Pascal PAS1 diskette in diskette drive B. You can now apply the update to the PAS1 diskette with the DOS DEBUG program on the DOS diskette.

The DEBUG program prompt is "-." All the things that you type are after the "-."

When you are responding to the "-" prompt, you must press the Enter key. Note that the *xxxx* in the data displayed line will be filled with the appropriate memory addresses:

```
A>DEBUG
-LDS: 100  1  A5  1
-DDS: 177  LA
xxxx:0177  41-3A 50 41 53 4B 45 59 20        A:PASKEY
xxxx:0180  20
```

If you do not see the above line of data after the "DDS:177 LA" command then you did something wrong and you should start again. You can start again by typing Q followed by Enter. This returns you to DOS:

```
-EDS: 177 'PASKEY  '
-WDS: 100  1  A5  1
-Q
```

(Remember to enter the two blanks after PASKEY.)

You are now back in DOS; you should now recreate your working PAS1 diskette from the updated diskette and mark both diskettes so you know which are the updated ones. The PAS1.EXE file on the updated PAS1 diskette will now look for the PASKEY file in the current directory on the default drive.

Appendix D. Considerations for Using Applications

If you have any of the following applications, please refer to the appropriate section in this appendix for additional information about using these applications with DOS 2.10.

> **Note:** This information is for users of the IBM Personal Computer and the IBM Personal Computer XT. For any other model IBM Personal Computer, see your dealer about using these application programs with DOS 2.10.

- Accounting Packages by BPI Systems, Inc.
- Accounting Packages Version 1.00 by Peachtree Software, Inc.
- Accounting Packages Version 1.10 by Peachtree Software, Inc.
- Arithmetic Games 1 and 2
- Asynchronous Communications Support Version 1.00
- Asynchronous Communications Support Version 2.00
- EasyWriter Version 1.10
- Fact Track
- Multiplan

- pfs:FILE
- pfs:REPORT
- SNA 3270 Emulation and RJE Support Version 1.00
- The Dow Jones Reporter Version 1.00
- Typing Tutor
- VisiCalc
- 3101 Emulator Version 1.00

Accounting Packages by BPI Systems, Inc.

These packages are designed for, and should be used only with DOS 1.00 or DOS 1.10. You should not use these applications with DOS 2.10.

Accounting Packages Version 1.00 by Peachtree Software, Inc.

These packages are designed for, and should be used only with DOS 1.00 or DOS 1.10. You should not use these applications with DOS 2.10.

Accounting Packages Version 1.10 by Peachtree Software, Inc.

These packages are designed to be used with either DOS 1.10 or DOS 2.10. If you use these packages with DOS 2.10 on a diskette-only system, you must have 128K bytes memory and 320K byte diskette drives in order to accommodate the larger size of DOS 2.10 and BASIC 2.00.

The Accounting Packages Version 1.10 by Peachtree Software, Inc. include functions allowing both programs and data files to reside on the IBM Personal Computer fixed disk. When using the fixed disk, these packages require DOS 2.10 and 128K bytes of memory.

Data files created using Accounting Packages Version 1.00 by Peachtree Software, Inc. may be used with the Version 1.10 packages.

Arithmetic Games 1 and 2

When using the arithmetic games with DOS 2.10, you must have a minimum of 96K bytes of memory. In addition, if you have a single-drive IBM Personal Computer, you should use the single-drive setup procedure.

Asynchronous Communications Support Version 1.00

To use this application with DOS 2.10 you must have a minimum of 96K bytes of memory.

To install this application in a subdirectory on a fixed disk, use the following procedure. This procedure assumes the following:

- The name of the subdirectory is ASYNC.
- The fixed disk is the default drive.
- The fixed disk drive is drive C.
- The root directory of the fixed drive contains DOS 2.10.

The Procedure

1. Insert the Asynchronous Communications Support diskette in drive A.

2. Enter the following commands in order to create the subdirectory ASYNC and to copy the files from drive A to drive C:

   MD \ASYNC
      Create the subdirectory

   CD \ASYNC
      Operate from it

   COPY A:*.* C:\ASYNC
      Copy all files on the application diskette

   COPY C:\BASIC.COM C:\ASYNC
      Copy BASIC.COM to the subdirectory

   ERASE UPDATE.BAT
      Not needed

   ERASE MESSAGE
      Not needed

3. You must now patch TERMINAL.BAS by doing the following:

BASIC	Run BASIC
LOAD "TERMINAL.BAS	Load the program to be patched
1 DEFINT I-M:DIM M(12):I=0:J=0:K=0:DATA &H55,&H8B,&HEC,&H8B, &H7E,&H06,&H8C,&HD8, &H05,&H00,&H10,&H89, &H05,&H5D,&HCA,&H02, &H00:I=0:J=0:K=0: IDSEG=o	Patch 1
2 J=VARPTR(M(1)):FOR I=1 TO 17:READ K:POKE I+J−1,K:NEXT:ON ERROR GOTO 3: J=VARPTR(M(1)):CALL J(IDSEG):ON ERROR GOTO 0:GOTO 4	Patch 2
3 RESUME 4	Patch 3
4 ON ERROR GOTO 0:ERASE M:CLS:LOCATE 10,5:PRINT "Initializing": I=0:J=0:K=0:DEF S EG=IDSEG	Patch 4
210 DEF SEG =IDSEG	Patch 5
115	Delete this line
205 (DEFINT I−M)	Delete this line
117 D$(1)="C"; DS$(2)="A"	Patch for Asynch 1.00 with PC-XT fixed disk
SAVE "TERMINAL.BAS"	Save the patched program
SYSTEM	Leave BASIC and return to DOS 2.10

4. You may now run the Asynchronous Communications Support program by doing the following:

CD \ASYNC	Operate from the subdirectory
AUTOEXEC	Run the program
CD \	Return to the root directory

Notes:

1. The use of the subdirectory name of ASYNC is an example only. You may use any name you wish.
2. All file transfers will be to and from the files in the subdirectory unless the printer is the destination.
3. This program tests the length of a filespec, and does not accept subdirectory names as part of a filespec.
4. You may rename AUTOEXEC.BAT if you choose.

Asynchronous Communications Support Version 2.00

To use this application with DOS 2.10 you must have a minimum of 96K bytes of memory.

To install this application in a subdirectory on a fixed disk, use the following procedure. This procedure assumes the following:

- The name of the subdirectory is ASYNC.
- The fixed disk is the default drive.
- The fixed disk drive is drive C.
- The root directory of the fixed drive contains DOS 2.10.

The Procedure

1. Insert the Asynchronous Communications Support diskette into drive A.

2. Enter the following commands in order to create the subdirectory ASYNC and to copy the files from drive A to drive C:

   MD \ASYNC
   Create the subdirectory

   CD \ASYNC
   Operate from it

   COPY A:*.* C:\ASYNC
   Copy all files on the application diskette

   COPY C:\BASIC.COM C:\ASYNC
   Copy BASIC.COM to the subdirectory

   ERASE UPDATE.BAT
   Not needed

   ERASE MESSAGE
   Not needed

3. You may now run the Asynchronous Communications Support program by doing the following:

   CD \ASYNC
   : Operate from the subdirectory

   AUTOEXEC
   : Run the program

   CD \
   : Return to the root directory

4. You may run the file conversion program, FILECONV, by doing the following:

   CD \ASYNC
   : Operate from the subdirectory

   FILECONV
   : Run the FILECONV program

   CD \
   : Return to the root directory

   Notes:

   1. The use of the subdirectory name of ASYNC is an example only. You may use any name you wish.

   2. All file transfers will be to and from the files in the subdirectory unless the printer is the destination.

   3. This program tests the length of a filespec, and does not accept subdirectory names as part of a filespec.

   4. You may rename AUTOEXEC.BAT if you choose.

EasyWriter Version 1.10

To use EasyWriter Version 1.10 with DOS 2.10 it is recommended your IBM Personal Computer have a minimum of 128K bytes of memory.

To copy DOS 2.10 to your EasyWriter program diskette, use the following procedure:

1. Follow the instructions in the "Startup" section of your EasyWriter manual for copying DOS to your EasyWriter program diskette. While copying DOS 2.10 to the program diskette, you may see the message:

   INSUFFICIENT DISK SPACE (or)

   ```
   SECTOR NOT FOUND  ERROR WRITING DRIVE B
   ABORT, RETRY, IGNORE?
   ```

   If you see the second message above, enter I (for ignore).

   > Note: For a system with a fixed disk, follow the instructions for copying DOS 2.10 to your EasyWriter program diskette for a one-drive system.

Due to insufficient diskette space, the DOS 2.10 FORMAT.COM utility was not transferred to your EasyWriter diskette. To use the FORMAT.COM utility, insert the DOS 2.10 diskette in drive A and a blank diskette in drive B. Then type: FORMAT B: and press the enter key. This will format the diskette in drive B.

2. EasyWriter 1.10 recognizes drive A and drive B as the drives to read or write from for EasyWriter files. Because the fixed drive is initially defined as drive C, you must reassign its name to be drive B. Enter:

   ```
   ASSIGN B=C
   ```

3. You can have your fixed disk automatically assigned to be drive B everytime you load your EasyWriter program. Use the following procedure to do this:

   a. Copy DOS 2.10 onto your EasyWriter program diskette following the instructions given above.

   b. Start DOS 2.10 from the fixed disk. The DOS prompt C> should appear.

   c. Put your EasyWriter program diskette into drive A.

   d. Enter COPY ASSIGN.COM A:

   e. Enter COPY CON A:AUTOEXEC.BAT

   f. Enter DATE

   g. Enter TIME

   h. Enter ASSIGN B=C

   i. Enter EW

   j. Press the F6 key. Now press the Enter key.

   The fixed disk can now be used to store and read your EasyWriter data files. Every time you load your EasyWriter program the fixed disk will automatically be assigned as drive B.

Fact Track

To use Fact Track with DOS 2.10 you need 96K bytes of memory.

If you have DOS 1.10, you should use it to follow one of the setup procedures documented in the Fact Track user manual. The setup procedure copies the necessary DOS programs onto the Fact Track program diskette. From then on, start the Fact Track program diskette in drive A.

If you have only DOS 2.10, and if you have an IBM Personal Computer with one or two disk drives, you should start up DOS in drive A. Enter the date and time as requested. When the system responds:

```
A>
```

you enter:

```
BASICA
```

When the system responds:

```
OK
```

remove the DOS diskette from drive A and insert the Fact Track diskette into drive A and close the door.

Enter:

```
RUN "COLOR
```

Do this every time you want to run the Fact Track program.

If you have DOS 2.10, but do not have DOS 1.10, and an IBM Personal Computer with a fixed disk that contains DOS or BASICA, you should start up DOS from the fixed disk. Enter the date and time as requested. When the system responds:

```
C>
```

put the Fact Track program diskette in drive A and close the door.

You enter:

```
A:
```

When the system responds:

```
A>
```

you enter:

```
C:BASICA COLOR
```

Do this every time you want to run the Fact Track program.

Multiplan

Please take note of this warning if you are planning to use DOS 2.00 or DOS 2.10 on the Multiplan Program Diskette. Due to space limitations on the Multiplan Program diskette and the increase in size of DOS 2.00 and 2.10, only DOS 1.1 or earlier versions may be placed on the program diskette. Do NOT attempt to copy any DOS other than DOS 1.1 or earlier versions to your Multiplan Program diskette.

Warning: Do not attempt to use DOS 2.00 or DOS 2.10 with Multiplan or you may permanently damage your Multiplan Program diskette.

If you have any further questions please see your nearest IBM Personal Computer dealer for additional information.

pfs:FILE

To use pfs:FILE with DOS 2.10 you need a minimum of 128K bytes of memory.

Follow the instructions in Part 1 of the Introduction "Getting Ready to Use FILE." Use your DOS 2.10 diskette whenever the instructions ask for the DOS diskette.

Using pfs:FILE with the IBM Fixed Disk

You can use pfs:FILE with the IBM fixed disk in two different ways. First, you can store your data files on the fixed disk, thus allowing larger files to be stored, and faster access to forms in the stored files. Second, you can copy the pfs:FILE program to the fixed disk and then load it from there. This allows you to load the program faster, without using the program diskette.

Storing PFS Data Files on the Fixed Disk

You can store a data file on the fixed disk provided you include the drive identifier for the fixed disk as part of the file name. If you specify the fixed disk drive as the default drive, then it is not necessary to include the drive identifier as part of the file name. Note that your data file must be in the same directory as your pfs:FILE program.

Copying pfs:FILE to the Fixed Disk

Follow these steps to copy the pfs:FILE program to the fixed disk:

1. Insert the DOS 2.10 diskette into drive A and turn on your IBM Personal Computer. Enter the date and time when the computer asks you to do so.

2. When the DOS prompt appears, remove the DOS 2.10 diskette and replace it with the pfs:FILE program diskette.

3. Follow the instructions in Appendix D of the pfs:FILE manual called "Setting Up a Serial Printer and the Work Drive" to run the setup program. Change the work drive name to the drive name of your fixed disk. For example, if your fixed drive is drive C, then the work drive item on the setup menu should be drive C.

4. If you have a serial printer, enter the correct values for the other items on the setup menu.

5. Press the F10 key to complete the setup program. Then enter FTRANS in response to the DOS prompt.

The in use lights will come on alternately as the program is copied from the diskette to the fixed disk. The copy is placed in the current directory. The DOS prompt reappears when the copy is complete.

If you have more than one fixed disk, the copy will be made to the drive whose name is last in the alphabet. For example, if two drives are named C and D, the copy will automatically be made to drive D.

Error Conditions

The following error conditions might occur during the copy procedure.

Message	Explanation	Corrective Action
CAN'T COPY PROGRAM FILE	Your program diskette has been damaged.	Try the copy procedure with the backup copy of the program diskette.
	Your fixed disk is improperly formatted. (also applies to the CREATE message)	Copy to diskette any files on the fixed disk. Then reformat the fixed disk, and try the copy procedure again.
CAN'T CREATE PROGRAM FILE	Your current directory is full.	If you have unnecessary files in the root level directory, delete them and start the copy procedure again.
NAME ERROR	Possible DOS error.	Reload DOS 2.10 and start the copy procedure again.
WRONG VERSION	Cannot find the fixed disk.	Make sure that your fixed disk is properly connected, and that you are using DOS 2.10.

Running the pfs:FILE Program from a Fixed Disk

To run the pfs:FILE program from a fixed disk, make sure that you have followed the instructions under "Copying pfs:FILE to the Fixed Disk." Then in response to the DOS prompt, type and enter the drive identifier for the fixed disk followed by the name of the program. For example, to run the pfs:FILE program from drive C, enter C:FILE. If the default drive is C, you need only enter FILE in response to the DOS prompt.

Changing Settings When Using the Fixed Disk

To change the work drive or the information stored for your serial printer after copying the pfs:FILE program to the fixed disk, you need to insert the pfs:FILE program diskette into drive A and run the setup program from that diskette. Use the COPY command to copy the file named IBMSETUP.PFS from the diskette in drive A to the fixed disk.

pfs:REPORT

To use pfs:REPORT with DOS 2.10 you must have a minimum of 128K bytes of memory, and you should do the following:

1. Copy the sample file called STAFF, which is on the pfs:REPORT program diskette, to a blank formatted diskette or to a fixed disk. See "Formatting Diskettes" and "Copying a File" in Appendix C of the pfs:REPORT manual.

2. Erase the STAFF file from the pfs:REPORT program diskette using the ERASE command of DOS 2.10.

3. Now follow the instructions in Part 1 of the Introduction in the pfs:REPORT manual "Getting Ready to Use REPORT." Use your DOS 2.10 diskette when the instructions ask for the DOS diskette.

Using pfs:REPORT with the IBM Fixed Disk

You can use pfs:REPORT with the IBM fixed disk in two different ways. First, you can store your files on the fixed disk, thus allowing larger files to be stored, and faster access to forms in the stored files. Second, you can copy the pfs:REPORT program to the fixed disk and then load it from there. This allows you to load the program faster, without using the program diskette.

Storing PFS Data Files on the Fixed Disk

You can store a data file on the fixed disk provided you include the drive identifier for the fixed disk as part of the file name. If you specify the fixed disk drive as the default drive, then it is not necessary to include the drive identifier as part of the file name. Note that your data file must be in the same directory as your pfs:FILE program.

Copying pfs:REPORT to the Fixed Disk

Follow these steps to copy the pfs:REPORT program to the fixed disk:

1. Insert the DOS 2.10 diskette into drive A and turn on your IBM Personal Computer. Enter the date and time when the computer asks you to do so.

2. When the DOS prompt appears, remove the DOS 2.10 diskette and replace it with the pfs:REPORT program diskette.

3. Follow the instructions in Appendix D of the pfs:REPORT manual called "Setting Up a Serial Printer and the Work Drive" to run the setup program. Change the work drive name to the drive name of your fixed disk. For example, if your fixed drive is drive C, then the work drive item on the setup menu should be drive C.

4. If you have a serial printer, enter the correct values for the other items on the setup menu.

5. Press the F10 key to complete the setup program. Then enter RTRANS in response to the DOS prompt.

   The “in use” lights will come on alternately as the program is copied from the diskette to the fixed disk. The copy is placed in the current directory. The DOS prompt reappears when the copy is complete.

   If you have more than one fixed disk, the copy will be made to drive whose name is last in the alphabet. For example, if two drives are named C and D, the copy will automatically be made to drive D.

Error Conditions

The following error conditions might occur during the copy procedure.

Message	Explanation	Corrective Action
CAN'T COPY PROGRAM FILE	Your program diskette has been damaged.	Try the copy procedure with the backup copy of the program diskette.
	Your fixed disk is improperly formatted. (also applies to the CREATE message)	Copy to diskette any files on the fixed disk. Then reformat the fixed disk, and try the copy procedure again.
CAN'T CREATE PROGRAM FILE	Your current directory is full.	If you have unneeded files in the current directory, delete them and start the copy procedure again.
NAME ERROR	Possible DOS error.	Reload DOS 2.10 and start the copy procedure again.
WRONG VERSION	Cannot find the fixed disk.	Make sure that your fixed disk is properly connected, and that you are using DOS 2.10.

Running the pfs:REPORT Program from a Fixed Disk

To run the pfs:REPORT program from a fixed disk, make sure that you have followed the instructions under "Copying pfs:REPORT to the Fixed Disk" (make sure the current directory contains pfs:REPORT). Then in response to the DOS prompt, enter the drive identifier for the fixed disk followed by the name of the program. For example, to run the pfs:REPORT program from drive C, enter: C:REPORT. If the default drive is C, you need only type and enter REPORT in response to the DOS prompt.

Changing Settings When Using the Fixed Disk

To change the work drive or the information stored for your serial printer after copying the pfs:REPORT program to the fixed disk, you need to insert the pfs:REPORT program diskette into drive A and run the setup program from that diskette. Use the COPY command to copy the file named IBMSETUP.PFS from the diskette in drive A to the fixed disk.

The Dow Jones Reporter Version 1.00

The Dow Jones Reporter Version 1.00 directs data to be saved only on diskette drive A. In addition, the program diskette is copy protected, and you cannot install the program on a fixed disk.

Dow Jones is a registered trademark of the Dow Jones Company, Inc.

SNA 3270 Emulation and RJE Support Version 1.00

To use SNA 3270 Emulation and RJE Support Version 1.00 with DOS 2.10 you need a minimum of 128K bytes memory.

To install this application in a subdirectory, SNA, on a fixed disk, use the procedure described below, which assumes the following:

- The fixed disk is drive C.
- The fixed disk drive is the default drive.
- The root directory in the fixed disk contains DOS 2.10.

The Procedure

1. Insert the SNA program diskette into diskette drive A.
2. Edit the files 3270COPY.BAT and SRJECOPY.BAT to change all drive B references to drive C.
3. Enter the following commands to create the subdirectory, SNA, and to copy the needed files from drive A to drive C.

   ```
   MD \SNA
       Create the subdirectory
   CD \SNA
       Operate from it
   A:3270COPY
       Copy required files
   ```

A:SRJECOPY
    Copy required files

COPY C:\BASIC.COM C:\SNA
    Copy a needed file

4. You may now run the SNA program by entering the following commands:

   C: Change the default drive

   CD \SNA
       Operate from the SNA subdirectory

   programname
       Enter the correct program name as specified in the SNA manual in response to the DOS prompt

   CD \
       Return to the root directory

   Notes:

   1. The use of the subdirectory name SNA is as an example only. You may use any name you wish.

   2. All file transfers for SRJE are to and from files in the SNA subdirectory.

   3. The SNA program does not accept subdirectory names as part of a filespec.

Typing Tutor

To use Typing Tutor with DOS 2.10 it is recommended that your IBM Personal Computer have a minimum of 64K bytes memory. If you have a single diskette drive system, you should use the single drive setup procedure.

VisiCalc by VisiCorp.

To use VisiCalc with DOS 2.10 it is recommended that your IBM Personal Computer have a minimum of 128K bytes of memory. The following are procedures to put DOS 2.10 on your VisiCalc program diskette, and to make your program diskette self starting for fixed disk. These procedures assume the following:

- You have at least one diskette drive and one fixed disk.
- You are familiar with loading DOS 2.10 from your fixed disk.
- You are aware that VisiCalc supports a maximum of two secondary storage devices. Only one of these two devices may be accessible during operations.

Putting DOS 2.10 on Your Program Diskette

Follow these steps to put DOS 2.10 on your VisiCalc program diskette and to make that diskette self starting:

1. Start DOS 2.10. You should see the DOS prompt C>.
2. Remove the write protect tab from the VisiCalc program diskette.
3. Place the program diskette into drive A.
4. Enter COPY COMMAND.COM A:
5. Enter SYS A:
6. Enter COPY ASSIGN.COM A:
7. Enter COPY CON A:AUTOEXEC.BAT

8. Enter DATE
9. Enter TIME
10. Enter ASSIGN B=C (or any other drive designator)
11. Type VC80
12. Enter ASSIGN (space) (press F6) (then press Enter)
13. Remove the program diskette and replace the write protect tab on your program diskette.

The fixed disk can now be used to store and retrieve your VisiCalc data files. Every time you load your VisiCalc program the fixed disk will be assigned as drive B.

3101 Emulator Version 1.00

To use the 3101 Emulator Version 1.00 with DOS 2.10 requires a minimum of 96K bytes memory.

Use the procedure described below to install the 3101 Emulator in a subdirectory, EM3101, on the fixed disk. The procedure assumes the following:

- The fixed disk is drive C.
- The fixed disk is the default drive.

The Procedure

1. Insert the 3101 Emulator program diskette into drive A.

2. Enter the following commands to create the subdirectory, EM3101, and to copy the needed files from drive A to drive C:

   MD \EM3101
   : Create the subdirectory

   CD \EM3101
   : Operate from it

   COPY A:*.* C:\EM3101
   : Copy all files on the 3101 Emulator program diskette

   ERASE COPYFILS.BAT
   : Not needed

   ERASE AUTOEXEC.BAT
   : Not needed

   CD \
   : Return to root directory

3. You may now run the 3101 Emulator by entering the following commands:

   CD \EM3101
   : Operate from the EM3101 subdirectory

   IBM3101
   : Run the 3101 Emulator program

   CD \
   : Return to the root directory

4. You may run the file conversion program, FILECONV, by entering the following commands:

   CD \EM3101
   : Operate from the EM3101 subdirectory

   FILECONV
   : Run the file conversion program

   CD \
   : Return to the root directory

   Notes:

   1. The use of the subdirectory name EM3101 is as an example only. You may use any name you wish.

   2. All file transfers will be to and from files in the EM3101 subdirectory.

   3. The 3101 Emulator does not accept subdirectory names as part of a filespec.

Appendix E. DOS Version 2.00 and 2.10 Enhancements

The information in this appendix is divided into two categories—those topics that apply to all users, and those topics that apply to system programmers or application developers. In each case, a brief description of the feature or change is offered, and you are referred to another section of the book for further details.

For All Users

DOS Versions 2.00 and 2.10 incorporate the following new and changed features:

- *Special characters.* The characters <, >, ¦, and \ now have special meanings to DOS, and can no longer be used in filenames. If you have files whose names contain any of these characters, they should be renamed (using your *old* version of DOS) before attempting to use them with DOS Version 2.00 or 2.10.

- *Configuration file.* You can create a file of special commands that DOS will read each time it starts up. The commands allow you to specify the number of disk buffers DOS should use, the names of device drivers, and additional information concerning DOS operation. Please refer to "Configuring Your System" in Chapter 4 for additional information.

✓

- *Support for one or more fixed disk devices.* The disk can be divided into multiple partitions, each usable by a different operating system. You can start (boot) your operating system from the fixed disk, and utility programs included to perform disk initialization, backup, and restore functions. If you have a fixed disk, please read the "Preparing Your Fixed Disk" information in Chapter 3 for setup instructions and the BACKUP and RESTORE commands in Chapter 2 for their respective functions.

- *Support for increased diskette capacity.* Beginning with DOS Versions 2.00 and 2.10, DOS formats diskettes at 9 sectors per track, which increases capacity from 163840 to 184320 characters of information for single-sided diskettes, and from 327680 to 368640 characters for dual-sided diskettes. The smaller capacity diskettes created by DOS Version 1.00 or DOS Version 1.10 (8 sectors per track) are also usable with DOS Versions 2.00 and 2.10. You do not need to reformat them. Please see the FORMAT and DISKCOPY commands in Chapter 2 for more information.

- *Multiple disk buffers.* A *disk buffer* is an area of user memory that DOS reserves at startup and is used for performing disk and diskette operations. DOS normally allocates two disk buffers at start-up time. Some users, however, will find that certain applications, such as data base applications, may run faster if DOS has more buffers available to it. DOS Versions 2.00 and 2.10 allow you to specify the number of buffers that DOS should reserve at start-up time. Please refer to "Configuring Your System" in Chapter 4 for instructions on specifying additional buffers.

- *Tree-Structured Directories.* This new feature allows you to place related groups of files in their own directories—all on the same disk. The individual directories are isolated from each other, giving the appearance of separate disks. Therefore, a search for a file in a given directory will not "see" files in other directories on the same disk.

  Each directory, beginning with the normal system directory (called the *root directory*) may contain special entries naming other directories on the same disk. These other directories, in turn, may contain entries for even more directories, and so on. When viewed in a logical order beginning with the root directory, the directory structure appears much like a diagram of a family tree—thus the term *tree-structured directories.*

  You may add or remove directories, copy files from one directory to another, instruct DOS to look in a specific directory to locate a file, etc. For complete details, please refer to Chapter 5 "Using Tree-Structured Directories".

- *Disk Volume Labels.* This feature allows you to specify a unique volume label (up to 11 characters) at the time you format a disk. The volume label is placed in the root directory, and is included in the displays produced by the DIR, CHKDSK, and TREE commands. Please refer to the FORMAT command for further information.

V

- *Extended DOS screen and keyboard control.* This feature allows you to issue special character sequences from within your program that DOS will use for screen cursor positioning and color, and further allows you to assign the meaning of any key on the keyboard. For example, you may assign the character string "DIR A:" to F10 so that simply pressing F10 has the same result as entering the **DIR A:** command. Please refer to "Configuring Your System" in Chapter 4 for more detailed information.

- *Redirection of Standard Input and Output.* This feature applies to all DOS programs that read from the keyboard or write to the screen (the standard input and output devices). By using the special characters < (for input) and > (for output), you can cause a program to receive its input from a source other than the keyboard, or to direct its output to a destination other than the screen. For example, the command:

  ```
  DIR A:>DIRLIST
  ```

  causes the directory listing from drive A to be placed in a file named DIRLIST on the default drive. Device names can also be used. For example, the command:

  ```
  DIR A:>PRN
  ```

  causes the directory listing to appear on the printer instead of the screen. Please refer to "Redirection of Standard Input and Output" in Chapter 1 for further information.

- *Piping of standard input and output.* This feature allows the standard output of one program to be used as the standard input to another. DOS acts as a "pipeline" to direct the output of the first program to the input of the second—thus, the *term* "piping." For further information and an example of its use, please refer to "Piping of Standard Input and Output" in Chapter 1.

New Commands

The following new commands have been added to DOS Versions 2.00 and 2.10. Please consult the command descriptions in Chapter 2 for further details and examples of their use.

ASSIGN

Allows you to reassign drive letters so that a request for a given drive can be routed to a different drive.

BACKUP

Backs up one or more files from a fixed disk to diskettes.

BREAK

Allows you to specify when DOS should check for a Ctrl-Break being entered at the keyboard. Normally, DOS only performs this check during screen, keyboard, printer, or auxiliary device operation. With this command, you can instruct DOS to check for Ctrl-Break whenever a program requests DOS to perform *any* function (such as disk operations). In this way, it is possible to "break out" of a program that performs few or no screen or keyboard operations (such as a compiler).

CLS

Clears the screen when used from a batch file or the keyboard.

CTTY

Allows you to define a different primary console, so that a remote terminal device can be used in place of the standard screen and keyboard. This command also reverses this assignment, to restore the keyboard and screen as the standard input and output devices.

ECHO, IF, FOR, SHIFT, GOTO

New subcommands provided to extend the flexibility of batch processing.

FDISK

Initializes and configures a fixed disk.

> **Note:** This command must be used before you use your fixed disk for the first time. Please refer to Chapter 3 "Preparing Your Fixed Disk".

GRAPHICS

Allows the Shift-PrtSc keys (display screen contents on printer) to print the image of a graphics display screen.

MKDIR, RMDIR and CHDIR

Create, remove, and inform DOS to use a directory other than the system directory.

PATH

Allows you to specify one or more paths of directory names that DOS will search if the command you have issued was not found in the current directory. They allow conditional execution of commands within a batch file by causing DOS to check for specified conditions.

PRINT

Prints a queue (list) of files on the system printer while you are using the system for other work.

PROMPT

Allows you to change the system prompt to a desired string.

RECOVER

Recovers a specific file that cannot be copied or otherwise used because of a defective spot on the disk that prevents the file from being read. This command also recovers multiple files when the directory has been damaged.

RESTORE

Restores one or more files from a diskette to a fixed disk.

SET

Allows you to enter keywords and parameters into a DOS "environment" that is accessible by commands and applications.

TREE

Displays the entire directory structure of the specified disk.

VER

Displays the DOS version number on the screen.

VERIFY

Instructs DOS to perform a verify operation (or to stop performing the verify) each time data is written to disk, until a new verify command is issued to turn the verify feature off. The verify operation increases assurance that the data was properly recorded on disk (that is, it can be read without error).

VOL

Displays the volume label of the disk in the specified drive.

Enhanced Commands

The following commands, that existed in DOS Version 1.10, have been enhanced for DOS Versions 2.00 and 2.10. For more detailed information, please consult the individual command descriptions in Chapter 2.

CHKDSK

Supports the fixed disk, the new and old diskette formats, and analyzes all directories on the volume. It also enables you to create files containing all sectors that were found to be allocated but were not associated with a file, so that you can recover "lost" data. An important feature is that, unlike Version 1.10 CHKDSK, it will take *no* corrective action on the disk being analyzed unless instructed to do so.

COMP

The file compare utility now allows multiple files to be compared. For example, you can compare all of the files on one disk with their counterparts on another disk. Also, COMP no longer prompts you to insert diskettes before comparing.

DEBUG

Now contains a command allowing you to enter assembly language statements that are assembled directly into memory.

DIR

Now displays the volume identification of the specified disk and clearly identifies entries that contain the names of other directories. It also displays the amount of available space left on the disk.

DISKCOPY and DISKCOMP

Supports the new 9-sector-per-track diskette format.

EDLIN

The line editor contains several new subcommands for more flexible management of source data. They include commands to copy and move lines, and to merge the contents of another file. The Replace and Search commands have been changed to begin their search at the current line plus one.

ERASE

Now requires you to press the Enter key after entering the Y/N response to the

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

message that appears when you instruct DOS to erase all of the files on a volume. This is intended to prevent accidental erasure of all files from the larger capacity devices supported by DOS Versions 2.00 and 2.10.

FORMAT

Now formats diskettes at 9 sectors per track (in DOS 1.00 and 1.10 diskettes were formatted at 8 sectors per track), allowing each new diskette to hold more data. It also allows you to specify a volume identification that is recorded in the disk's directory. Support for initializing a fixed disk is also included.

LPT2:, LPT3:, and COM2:

Now recognized as valid device names by DOS, and can be used in place of filenames.

For Programmers

Programmers may find the *IBM DOS Technical Reference* helpful.

- DOS Versions 2.00 and 2.10 include the ability to install your own device drivers for character or block-oriented devices.

- Three changes were made to internal functions that cause different results from those obtained on DOS Version 1.10:

    1. The function call (hex 1B) that previously returned a pointer to the file allocation table now returns a pointer to only the table's identification byte, for purposes of determing the disk type. All applications that use call hex 1B to obtain the file allocation table should be changed to use interrupt hex 25 to read the file allocation table directly from the disk. The file allocation table always begins at logical sector 1, and its size can be determined from the information returned by call hex 36. We recommend that you avoid using calls hex 1B and hex 1C.

    2. The mapping of logical sectors on dual sided diskettes has been rearranged to facilitate program loading and to improve system performance.

        This change allows DEBUG to load an entire file with a single L command.

        Applications that use interrupts hex 25 and hex 26 on multi-sided disks or diskettes may require modification to operate properly on DOS Versions 2.00 and 2.10.

3. Additional bits have been defined in the file attribute byte of the DOS disk directory. Programs that depended upon the file attribute byte being equal to zero if a file was not a hidden or system file may not work correctly.

- A new set of function calls has been made available to provide a wide variety of services. We suggest that systems programmers and application developers review all of Chapter 5 of the IBM *DOS Technical Reference* for details on these new functions.

Appendix F. Some Keys You Use with DOS

In addition to the keys you'd find on a typewriter, your keyboard has some special keys you'll use with DOS.

Before you use the special keys, please be sure to read the keyboard section of the *IBM Guide to Operations*.

DOS Editing Keys

Use the DOS editing keys to make corrections to commands and input lines as they are being entered. Note that the meaning of these keys can change if you alter their assignments through extended keyboard control.

The DOS editing keys are used to edit *within* a line. The Line Editor (EDLIN) program operates on *complete lines* within a file or document. When you are working with EDLIN and want to edit within a line, however, use the DOS editing keys. For more information about EDLIN, refer to Chapter 6.

> **Note:** Some word processing programs define special editing rules; therefore, the DOS editing keys may not work as described in this chapter. You can also define special editing rules when using the BASIC Program Editor while programming in BASIC.

Any line you enter from the keyboard is retained in an input buffer when you press Enter. The line is then made available to your program for processing.

Since the line remains in the input buffer, you can use that line as a *template* for editing purposes. The DOS editing keys operate on that copy of the line. You can repeat or change the line by using the DOS editing keys, or you can enter an entirely new line.

Here is a summary of the DOS editing keys, their functions, and their locations on the keyboard:

DOS Editing Key	Function
Del	Skips over one character in the template. The cursor does not move.
Esc	Cancels the line currently being displayed. The template remains unchanged.
F1 or →	Copies one character from the template and displays it.
F2	Copies all characters up to a specified character.
F3	Copies all remaining characters from the template to the screen.
F4	Skips over all characters up to a specified character. (F4 is the opposite of F2.)
F5	Accepts an edited line for continued editing – the currently displayed line becomes the template, but it is not sent to the requesting program.
Ins	Allows you to insert characters within a line.

Examples of Ways to Use DOS Editing Keys

The following examples show how you use the DOS editing keys with the Line Editor (EDLIN) program.

If you want to try these examples, you must use the EDLIN program. The EDLIN program is on your DOS diskette and is discussed in Chapter 6. You do not need to review the EDLIN chapter to complete these examples – just follow the steps provided.

Notes:

1. Because the DOS diskette shipped with your IBM Personal Computer is *write protected*, you cannot create the file used in the following examples on that diskette. You must use a *copy* of your DOS diskette to complete these examples.

2. In the following examples, to *enter* something means that you should *type* the information and then press the Enter key.

3. If you finish one or more of the following examples and you do not want to try the rest of the examples, go to "To Stop the Editing Session" at the end of this chapter.

To Start EDLIN

1. Insert your DOS diskette into drive A.

2. Create a file named EXAMPLES.

   If you want the EXAMPLES file to reside on the diskette in your default drive, enter:

   ```
   EDLIN EXAMPLES
   ```

   or

If you want the EXAMPLES file to reside on the diskette in another drive, you must specify the drive, as in:

```
EDLIN B:EXAMPLES
```

This command tells DOS to load the EDLIN program and create a file called EXAMPLES.

The following message and prompt will be displayed:

```
New file
*_
```

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

3. Now, enter the letter **I**.

   This tells EDLIN that you want to begin *inserting* lines in the file named EXAMPLES.

   The screen looks like this:

```
New file
*I
  1:*_
```

4. Type **This is a mailorder file.** on line 1 and press Enter.

5. Type **Editing is easy.** on line 2 and press Enter.

   You now have two lines of text in your EXAMPLES file.

6. Press the Ctrl-Break keys.

   Pressing Ctrl-Break will end the insert mode of operation and return you to the EDLIN prompt.

7. Enter the number 1.

   This tells EDLIN that you want to display line 1 on the screen.

   The screen should look like this:

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

You are now ready to begin the examples.

> **Note:** If you encounter any problems while trying these examples, press the Ctrl-Break keys. The EDLIN prompt will be displayed and you can start over.

Example 1

Let's delete the first two characters in the word **This** and then copy the remainder of the line.

1. Press the Del key twice to delete the first two characters.

2. Press F3 to copy the remainder of the line to the screen. The screen looks like this:

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

If you want to continue with the next example:

1. Press Ctrl-Break to return to the EDLIN prompt. (The changes you made to line 1 will not be saved.)

2. Enter the number 1.

Example 2

Now we'll change line 1; then, using Esc, we will cancel the change. A backslash (\) will be displayed to indicate that the displayed line has been cancelled.

Note: If the insert mode is on, the system automatically turns it off when you use Esc.

The screen looks like this:

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

To change line 1 to **Sample file:**

1. Type **Sample file**, but do *not* press Enter.

```
1:*This is a mailorder file.
1:*Sample file_
```

2. To cancel the line we just entered, press the Esc key.

```
1:*This is a mailorder file.
1:*Sample file \
   _
```

Now we can continue to edit the original line **This is a mailorder file.**

3. Press F3 to copy the original line to the screen.

The screen looks like this:

```
1:*This is a mailorder file.
1:*Sample file \
   This is a mailorder file._
```

If you want to continue with the next example:

1. Press Ctrl-Break to return to the EDLIN prompt.

2. Enter the number 2.

Example 3

Now let's copy one character by using F1 or →. (F1 or → is the opposite of Del. Del skips over one character in the template.)

The screen looks like this:

```
2:*Editing is easy.
2:*_
```

1. Press F1 or → key three times.

   The screen looks like this:

   ```
   2:*Editing is easy.
   2:*Edi_
   ```

   Each time you press F1 or → key, one more character appears.

If you want to continue with the next example:

1. Press Ctrl-Break to return to the EDLIN prompt.

2. Enter the number 2.

Example 4

Now let's use F2. Remember, F2 copies all characters from the template to the screen up to, but not including, the first occurrence of a specified character.

You must always specify a character when using this key. If the specified character is not present in the template, nothing is copied.

The screen looks like this:

```
2:*Editing is easy.
2:*_
```

1. Press F2 and enter the letter g.

   The screen looks like this:

   ```
   2:*Editing is easy.
   2:*Editin_
   ```

   Now we'll copy all the remaining characters in the template to the screen by using F3.

   (If you pressed Enter now, only **Editin** would be saved in the EXAMPLES file as line 2.)

2. Press F3.

   The screen looks like this:

   ```
   2:*Editing is easy.
   2:*Editing is easy._
   ```

If you want to continue with the next example:

1. Press Ctrl-Break to return to the EDLIN prompt.

2. Enter the number **1**.

Example 5

Now let's scan and locate specific characters within the template by using F4. This is a way to skip over characters. The cursor does not move when you use this key, and no characters are displayed.

You must always specify a character after you press F4. If the specified character is not present in the template, no characters in the template will be skipped.

We will also use F3 to copy the remaining characters in the template to the screen.

The screen looks like this:

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

1. Press F4 and enter the letter o. (No characters are displayed.)

2. Press F3 to copy the remainder of the line.

   The screen looks like this:

   ```
   1:*This is a mailorder file.
   1:*order file._
   ```

If you want to continue with the next example:

1. Press Ctrl-Break to return to the EDLIN prompt.

2. Enter the number 1.

Example 6

Now we'll move the currently displayed line into the template by using F5. Pressing F5 is the same as pressing Enter, except that the line is *not* sent to your program. An @ character is displayed to indicate that the new line is now the template.

> **Note:** If the insert mode is on, the system automatically turns it off when you use F5.

Once you press F5, you can continue to make changes to a line. When you are finished, press Enter to send the line to your program.

The screen looks like this:

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

1. Type This is not a sample file.

   The screen looks like this:

   ```
   1:*This is a mailorder file.
   1:*This is not a sample file._
   ```

2. Press F5.

   The result is:

   ```
   1:*This is a mailorder file.
   1:*This is not a sample file.@
      _
   ```

   The replacement line **This is not a sample file.** is now in the template. The replacement line is acceptable, but let's continue to edit it.

3. To remove the word **not** from the replacement line, press F1 eight times:

   ```
   1:*This is a mailorder file.
   1:*This is not a sample file.@
   ```

4. Press Del four times to remove one blank space and the word **not**.

5. Press F3 to copy the remaining characters to the screen.

   The screen looks like this:

   ```
   1:*This is a mailorder file.
   1:*This is not a sample file.@
      This is a sample file._
   ```

6. Press Enter to make the replacement line **This is a sample file.** the template in place of the original line *and* to send the line to your program.

   (If you want to do more editing without sending the line to your program, press F5 again to put the displayed line into the template.)

   > **Note:** Pressing Enter *immediately* after pressing F5 empties the template.

If you want to continue with the next example:

1. Press Ctrl-Break to return to the EDLIN prompt.

2. Enter the number 1.

Example 7

Let's look at an example using the Ins key. The Ins key serves as an on/off switch for entering and leaving insert mode. You can press the Ins key to enter insert mode, and press the Ins key again to leave the insert mode.

While in the insert mode of operation, any characters that you enter are *inserted* in the line being displayed. The characters do not *replace* characters in the template.

When you are not in the insert mode of operation, any characters that you enter *replace* characters in the template. If you are entering characters at the end of a line, the characters will be added to the line.

The screen looks like this:

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

Let's change the word **sample** to **salary**.

1. Press F2 and enter the letter **m**.

   The screen looks like this:

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

2. Press the Ins key and enter the characters **lary**.

   The screen looks like this:

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

   Notice that the characters **lary** were inserted, but no characters from the template were replaced.

3. Now, press Ins again to leave the insert mode.

4. Enter one blank space and the three characters **tax**.

   ```
   1:*This is a sample file.
   1:*This is a salary tax_
   ```

5. Press F3 to copy the remaining characters in the template to the screen.

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

   Notice that we *inserted* **lary** and we *replaced* **mple** with a blank space and **tax**.

6. Now press Enter to make the replacement line the template in place of the original line and send the line to the requesting program.

To Stop the Editing Session

You have now completed the examples.

To return to the A> prompt:

1. Press Ctrl-Break.

2. Enter the letter Q.

   Q tells EDLIN that you don't want to save the EXAMPLES file and that you want to *quit* the editing session. EDLIN will prompt you with this message:

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

   to make sure you don't want to save the file.

3. Enter the letter Y.

Index

IBM

The Personal Computer
Hardware Library

Reader's Comment Form

DOS 1502343

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Tape Please do not staple Ta

IBM

The Personal Computer
Hardware Library

Reader's Comment Form

DOS 1502343

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

USINESS REPLY MAIL

IRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Please do not staple

Tape

IBM

The Personal Computer
Hardware Library

Reader's Comment Form

DOS 1502343

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Please do not staple

Tape

IBM

The Personal Computer
Hardware Library

Reader's Comment Form

DOS 1502343

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Please do not staple

Tape

International Business Machines Corporation

P.O. Box 1328-W
Boca Raton, Florida 33432

1502343

Printed in United States of America