999-801-000IS

User's Guide

AT&T Personal Computer 6300

MS™-DOS By Microsoft®

Written by
Agora Resource, Inc.
Lexington, MA

NOTICE

The information in this document is subject to change without notice. AT&T assumes no responsibility for any errors that may appear in this document.

Contents

1 Introduction

2 Before You Start MS-DOS

3 Starting MS-DOS

4 Using MS-DOS Commands

5 MS-DOS Commands

6 Batch Processing Commands

Appendices

Index

1 Introduction

- What Is MS-DOS?
- What Is Covered in This Guide?
- How to Enter Keystrokes
- What Is Included with This Guide?

You may have already seen some of the material in this Guide in the *User's Guide*. If you have not read Chapter 2, **Getting Started,** and Chapter 4, **What Every User Should Know,** it is a good idea to familiarize yourself with them. They provide a useful introduction to MS-DOS.

This Guide gives you the concepts and rationale behind the MS-DOS procedures you followed in the *User's Guide*. It presents the material in the natural sequence you follow when using MS-DOS.

*MS™-DOS is a trademark of Microsoft®

What Is MS-DOS?

MS-DOS stands for Microsoft-Disk Operating System. An operating system is the traffic director of your computer. Like the signals at a busy intersection directing cars, the computer's operating system is at the center of activity directing data. MS-DOS is the operating system designed to run your AT&T computer. It coordinates the flow of information between the screen, keyboard, memory, and storage disks. It also directs other devices attached to your computer, such as a printer or telephone coupler.

You can make your computer perform certain tasks with a set of MS-DOS commands. This guide teaches both the fundamental MS-DOS commands and the more advanced commands that will extend your computing power.

What Is Covered in This Guide?

Chapter 1 explains how to follow the examples in this Guide. It also contains a checklist of the diskettes containing MS-DOS and a list of the other manuals to help you use your computer.

Chapters 2 and 3 introduce you to the basic concepts of MS-DOS and some of the important practices you should follow when working with a computer. You will learn about diskettes and files, and the vital importance of making backup copies of your important program and data diskettes. For users of a fixed disk drive system, Chapter 3 helps you get MS-DOS started on the fixed disk.

In Chapter 4, the concepts you've learned are put into practice. Contained here is a step-by-step introduction to the most commonly used MS-DOS commands, some information about automatic processing, and a recap of some useful practices.

Chapters 5 and 6 are reference chapters. They contain alphabetic listings of each of the MS-DOS commands, their purpose, correct syntax, rules of usage, and examples that illustrate how each command is used.

At the end of this Guide are Appendices which contain information about messages that appear on your screen, EDLIN, the line editor program on your MS-DOS/GWBASIC System diskette, and some information for users of a single diskette drive system.

How to Enter Keystrokes

Later in this guide you will be provided with step-by-step procedures to follow. You are instructed to type certain words and characters and to press specific keys. It is important to type exactly what is shown in the indented example (only the boldface part):

A>**type this**

The specific keys or sequence of keys which you should press appear like this:

RETURN

or

CTRL **NUM LOCK**.

In a sequence of two or more keys, as shown above, press and hold the first keys while you press the last key in the list.

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

What Is Included with This Guide?

This MS-DOS Version 2.11 Release 2.0 User's Guide comes with the following:

- MS-DOS/GWBASIC System diskette
  This diskette contains MS-DOS and its related programs.

- MS-DOS Supplemental Programs diskette
  This diskette contains the LINK Utility, EXE2BIN, DEBUG, BACKUP, SHIP and RESTORE.

Both diskettes are in the plastic pocket at the back of this guide. The MS-DOS System diskette referred to throughout this guide is the MS-DOS/GWBASIC System diskette.

2 Before You Start MS-DOS

About Diskettes

Caring for Diskettes
Write-Enable Notch
Why Copy Your MS-DOS System Diskettes?
Diskettes Need to Be Formatted

What System Do You Have?

If You Have a Single-Diskette Drive System
If You Have a Two-Diskette Drive System
If You Have a Fixed Disk Drive and a Diskette Drive System

Files

Naming Files
Reserved Filenames and Extensions
Wild Card Characters
File Specifications

About Diskettes

This chapter introduces you to some of the important concepts and practices you must know before you start using MS-DOS. If you are unfamiliar with using a computer and diskettes, the section of this chapter **About Diskettes** must be read carefully. If you have experience with computers, the practices in this section are second nature to you. New and experienced users alike should read the **Files** section, as the information here describes the file naming rules needed to use MS-DOS.

The following sections cover the basic information you need to know about diskettes. If you are new to using computers, you should read these sections carefully.

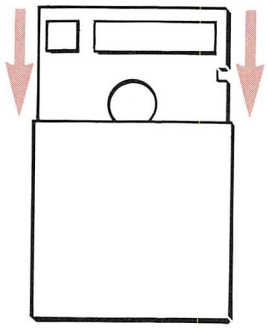
Caring For Diskettes

Your diskettes are very valuable. They contain information and data representing hundreds of hours of work. You must take care in their handling as well as follow prudent back up and archival practices.

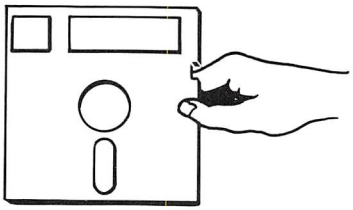
Here are some rules and helpful hints to follow.

- Always keep the diskette in its envelope when it's not in use. The envelopes are specially

treated to resist static electricity and dust accumulation.

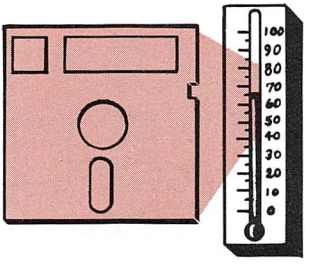


- Never touch the surface of the diskette. Handle the diskette by its protective cover.

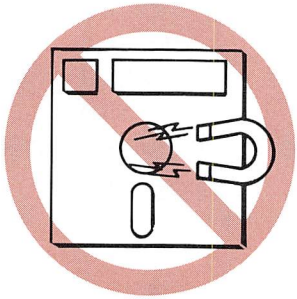


- Never bend or fold a diskette.

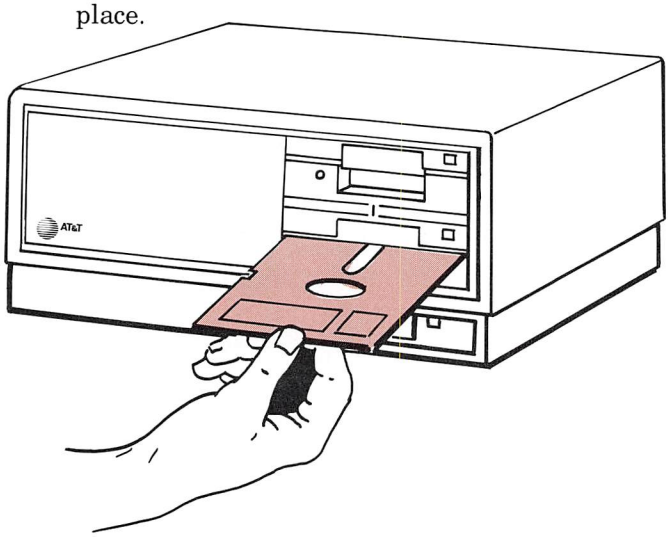
- Store and use diskettes in a safe environment. Don't let them get too cold or too hot. Keep them out of the sun or the trunks of automobiles.



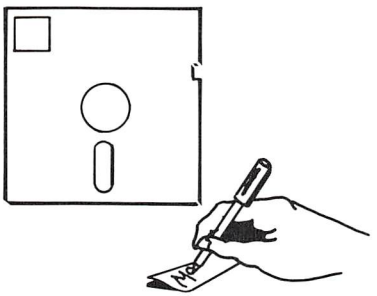
- Keep diskettes away from magnets or strong electrical fields. Keep diskettes away from your telephone. The information stored on the diskette can be damaged or erased.

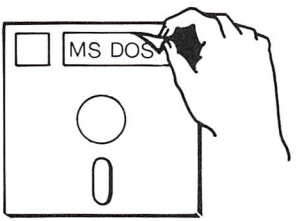


- Always be very careful when inserting the diskette into a drive. Push gently. Never force a diskette into place.

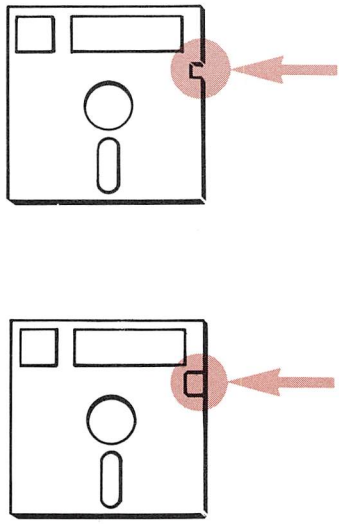


- Don't write on a diskette with a ball-point pen or pencil. Use only a soft felt-tipped marker. Better yet, prepare the diskette's label first, then apply it to the diskette.





Write-Enable Notch



Notice the small notch on a diskette below the label on the right. This is called the write-enable notch. If you cover the write-enable notch, information can only be read from the diskette. Data already on the diskette cannot be accidently erased or written over.

You can cover the notch with a tab supplied with the diskette or with a piece of opaque tape. You can remove the tape when you want to add or erase information.

It's a good idea to protect your important diskettes this way.

Many of the program diskettes you may buy have no notch. Such notchless diskettes are already "write-protected." The computer cannot write any information on a write-protected diskette.

Why Copy Your MS-DOS System Diskettes?

Your MS-DOS System diskette holds all of the programs you need to make your computer operate. To protect it, make a "working" copy and save the original, or Master diskette, in a safe place. You should only use the Master diskette to make additional working copies.

This is very important.

Diskettes can be lost, physically damaged, and, on occasion, accidentally erased by sudden electrical surges in your computer or power systems. One of the first things you should do is make "working" copies — duplicates — of your MS-DOS System diskettes.

These copies become your day-to-day working copies of valuable master diskettes.

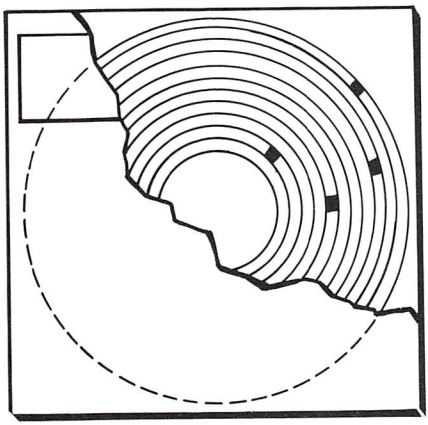
Be sure to:

- Use a felt-tip pen to label and date each working copy.
- Store the original or master diskettes in their paper envelopes in a safe place.
- Use the working copies in your daily work.

Diskettes Need to Be Formatted

Before you can use a brand new or blank diskette with MS-DOS, you must FORMAT the diskette. You need only do this once — before you use a diskette for the first time.

FORMATing a diskette prepares it to receive and store data. Think of it this way. The FORMAT command is like putting lines on a newly paved parking lot so that cars can be parked in an orderly and efficient way. The FORMAT command writes "lines" (track and sector markers) on a new diskette. Data then is placed on the diskette in an orderly way and can be easily found and read.



What System Do You Have?

Your computer system unit operates with one of the following:

- a single diskette drive
- two diskette drives
- a fixed (non-removable) disk and a diskette drive.

In this book you learn how to use MS-DOS with these configurations. If you have a one-diskette drive system, a two-diskette drive system, or a fixed disk and a diskette drive system, the information you need to know is in this chapter.

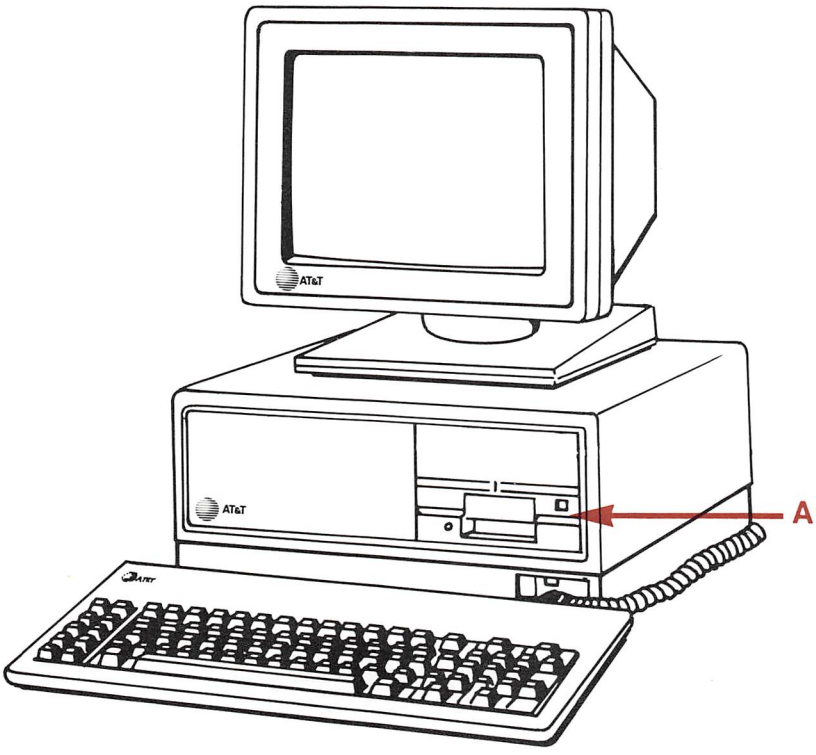
In our examples, we use the following conventions to distinguish between the drives in a two-drive system. These are common synonyms, used by the computer industry and by users alike.

Drive A (or diskette A) is also called:		**Drive B (or diskette B) is also called:**
Source	<———>	**Target**
Original	<———>	**Backup**
Master	<———>	**New or Blank**
First	<———>	**Second**

If You Have a Single-Diskette Drive System

If you have a single diskette drive, identify the drive as drive A.

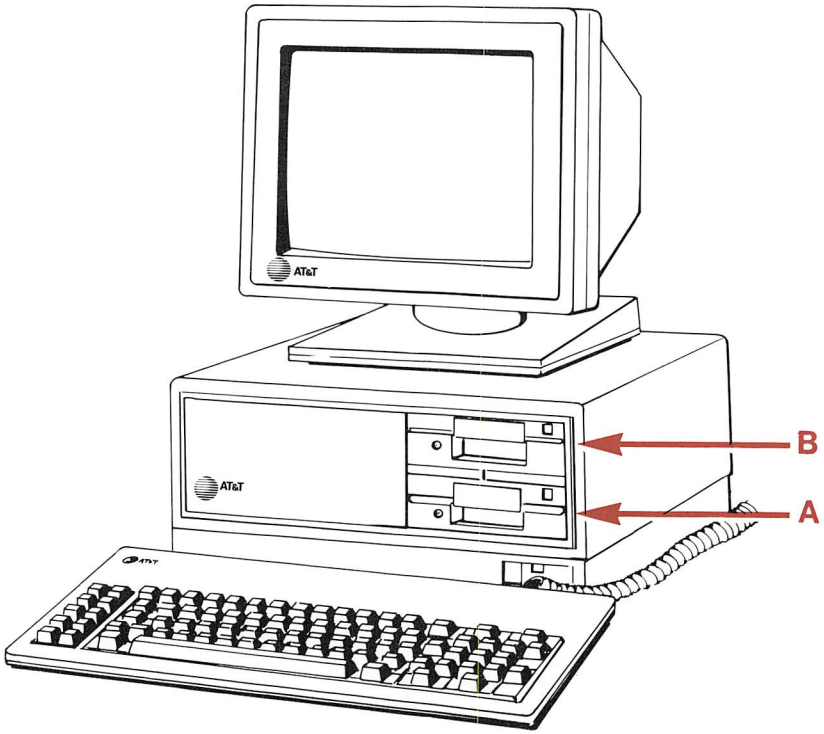
If MS-DOS tells you to insert a diskette into drive A, and then later to insert a diskette into drive B, put the first diskette into drive A, then take it out and put the second diskette into drive A.



If You Have a Two-Diskette Drive System

If you have two diskette drives, identify them as drive A (the bottom drive) and drive B (the top drive).

If MS-DOS tells you to insert a diskette into drive A and, later, a diskette into drive B, you can insert diskettes into both drives. You don't have to exchange diskettes if you have two diskette drives.



If You Have a Fixed Disk Drive and a Diskette Drive System

A fixed disk is treated just like a diskette for most of the MS-DOS commands. Like a diskette, a fixed disk has a drive designation letter. You can read and write data from and to it. A fixed disk can hold millions of characters of information and can retrieve data much faster than data can be retrieved from a diskette drive.

Some MS-DOS commands are used only with a fixed disk drive. These are covered in Chapter 5, MS-DOS Commands.

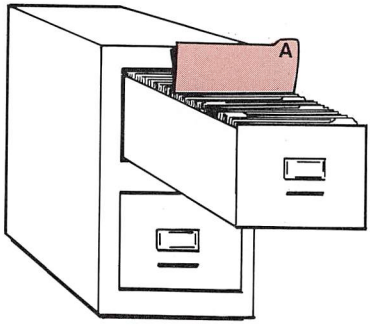
When you are copying the contents of a diskette to a fixed disk, the diskette drive is identified as drive A and the fixed disk is called drive C.

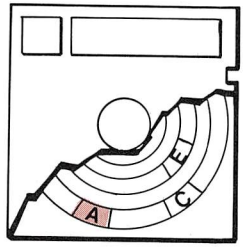
More information about setting up a fixed disk drive is found at the end of this chapter.

Files

Information is stored in files on your diskettes or fixed disk. A computer file is like a file folder in a conventional filing cabinet. It contains related data on a specific subject.

Every file has its own filename. MS-DOS uses those filenames to locate stored files.





Naming Files

Each file on a diskette must have its own unique filename. This way your computer can find the file you want.

You **can** use the same filename on two different diskettes or a fixed disk, but this may lead to problems later when you are COPYing a file from one disk to another.

A filename is composed of:

- a **name** of one to eight characters
- an **extension** of one to three characters (an extension is an optional identifier)
- a **period**, required as separator if an extension is used.

Legal characters in a filename and extension are:

- letters of the alphabet
- numbers 0 through 9
- these special characters:
  $ # & @ ! % ( ) - { ' ` ~ ^ _ }

NOTE: Spaces between characters are not allowed.

Examples of legal filenames are:

PHDTHSIS.78
MOM&SIS.LTR
$$TAXES$.83
CH3PART1
##4^.!!!

Although the last filename on this list is legitimate, it doesn't give you much information about the contents of the file. A good filename reminds you what information is in the file.

To see what files are on a diskette, use the MS-DOS DIR command. Files are displayed on the screen similarly to this:

```
FC          EXE   2585    4-12-84    9:00a
FDISK       COM   4640    4-12-84    9:00a
FIND        EXE   6331    4-12-84    9:00a
FORMAT      COM   5776    4-12-84    9:00a
.
.
.
RESTORE     COM   4043    4-12-84    9:00a
      33 File(s)          150528 bytes free

A>
```

Some of the names and numbers may be different, but the screen should show the same five columns and last line.

NOTE: On the line before the screen prompt, the number of files and free bytes will vary.

This list of files will become very familiar. You see this list on your screen every time you enter a DIR command for an MS-DOS System diskette.

The files on the MS-DOS System diskette are the programs that control your computer and perform important "housekeeping" tasks on your diskettes and files. Files on your diskettes contain many different types of information. There are simple text and data files — files that contain information that you create with a word processing or spreadsheet program. Other kinds of files are program files. These contain the instructions used by your computer to perform a task or complex operation. These are created using a programming language such as BASIC or Pascal — or other, specialized languages — and are meant to be used by a computer, rather than read by a person.

The files on your MS-DOS System diskette are program files.

This chart describes each of these MS-DOS System files and what they do.

Filename	Function of File
Contents of MS-DOS/GWBASIC System Disk	
ANSI.SYS	Lets you use standard terminal escape sequences
ASSIGN.COM	Routes requests for one drive to another
AUTOEXEC.BAT	File of commands executed automatically at system start-up
BASICA.COM	Loads GWBASIC into memory
BASIC.EXE	Loads GWBASIC into memory
CHKDSK.COM	Checks disks
CHMOD.COM	Changes/displays file and/or directory attributes
COMMAND.COM	Processes MS-DOS commands
COMP.COM	Compares files
DISKCOMP.COM	Compares diskettes
DISKCOPY.COM	Copies diskettes
EDLIN.COM	Line editor
FC.EXE	Compares files
FDISK.COM	Partitions a hard disk
FIND.EXE	Finds a string in a list of files or standard input
FORMAT.COM	Formats disks
GRAPHICS.COM	Sets up printer to print graphics

Filename	Function of File
GWBASIC.EXE	GWBASIC interpreter
MODE.COM	Sets display, communications, and serial printer environments
MORE.COM	Pages through text
PRINT.COM	Print spooler
RAMDISK.DEV	Device driver that lets you use RAM for file I/O, resulting in faster access
RECOVER.COM	Recovers files on disk
SIZE.COM	Lists filenames and file sizes of specified files
SORT.EXE	Sorts text
SYS.COM	Transfers MS-DOS system files from one disk to another
TREE.COM	Displays directories and their contents

Contents of Supplemental Disk

BACKUP.COM	Backs up files
DEBUG.COM	Debugger
EXE2BIN.EXE	Converts .EXE files to .BIN files
LINK.EXE	Linker
RESTORE.COM	Restores files saved with BACKUP
SHIP.EXE	Prepares hard disk for shipping

Even though the period is not displayed on the screen, you must use it when entering a filename and extension and when telling MS-DOS about that file.

Reserved Filenames and Extensions

Some filenames and extensions have a special meaning in MS-DOS. Some filenames identify the hardware parts of your computer or its accessories. Some extensions identify types of files handled in a special way by your computer.

Do not use the following reserved filenames when naming your own files.

Filename	Meaning
AUX	Refers to input or output to an auxiliary device — a printer or a modem
CON	Refers to keyboard input or displayed output
PRN	Refers to the printer
NUL	Used when you do not want to create a particular file, but requires an input or output filename.

Some of the special filename extensions that you may use from time to time identify program files that are used by the computer to perform some operation. For example, if you write a batch processing program that performs a series of MS-DOS commands, you use the extension .BAT to tell MS-DOS how to handle the file whenever you want to run that program. The extensions .COM and .EXE also tell MS-DOS that files are executable.

Wild Card Characters

When you are using MS-DOS file commands, the wild card characters * and ? can speed things up, particularly when you are working with multiple files. These characters provide flexibility in making choices about filenames and extensions.

The ? character indicates that any valid character may occupy that position in a filename or extension. For example:

```
DIR  INV???.84
```

lists all directory entries that begin with INV, have up to three following characters, and end with the extension 84. For example, these files might be found:

```
INV.84
INV003.84
INVOIC.84
```

The * character in a filename or extension indicates that any valid character can be in that position, all remaining positions, or in the extension. For example:

```
DIR  *.DAT
```

lists all the files with the extension .DAT. These files might be found:

```
84.DAT
INVENTRY.DAT
PAUL.DAT
Q184.DAT
```

The wild card characters * and **?** can be used together or interchangeably. They are powerful tools and should be used **very carefully**, particularly when using the DELete or ERASE commands.

File Specifications

When you want to call up a file, you must tell MS-DOS where to search for it — that is, you must specify what drive contains the diskette with that file. Type in:

```
A>type MOM&SIS.LTR
```

Here, you provided only the filename and extension. Therefore, MS-DOS searches the current A drive.

These three parts — the drive letter, the filename, and the extension — are called the file specification.

Often, you need to fetch a file from a drive other than the current drive. In these cases you need to specify the drive.
For example, to specify a file on the B drive with a current A drive, type:

```
A>type B:thisfile.onB
```

If your hard disk is the current drive, you may specify a file on A by typing:

```
C>type A:thisfile.onA
```

The drive letter and the colon are called the drive specifier. Always type the colon (:) after the drive letter. Do not put any spaces between the three parts of the file specification.

3 Starting MS-DOS

Loading MS-DOS

In this chapter you will learn about how to start your computer with MS-DOS and some of the basic techniques you will need to work with your operating system. Also, if you have a fixed disk drive in your computer, this chapter contains information about setting up MS-DOS on a fixed disk and some tips on using this type of disk drive.

If MS-DOS is already installed on your fixed disk, some of the information in this chapter may be familiar to you. Review each section in this chapter to be sure that you understand how to begin using MS-DOS with your computer.

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

Internal commands are placed into memory whenever you start your computer with MS-DOS. Internal commands are the MS-DOS operations you use most often. For example, DIR is an internal command.

External commands are the MS-DOS files on the System diskette placed in your computer's memory when you specifically need them to do some task. For example, the CHKDSK command is an external command.

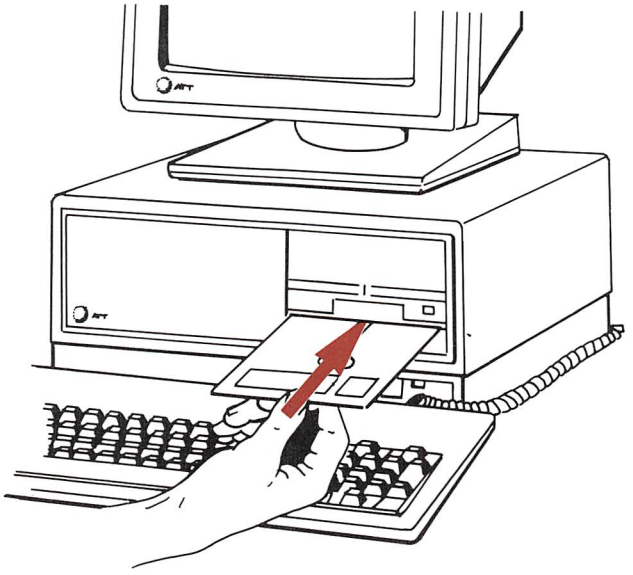
When you load or start MS-DOS, the MS-DOS internal commands are read from the COMMAND.COM file on the MS-DOS System diskette and placed into the computer's memory.

After this process is completed, the MS-DOS prompt A> or C> appears and you can enter a command.

Inserting Diskettes

To insert the diskette properly:

1. Remove the diskette from its paper sleeve.
2. Open the diskette drive door.
3. Slide the diskette into drive A with the label side up. Do not force or bend the diskette.
4. Shut the drive door carefully.



If Your Computer Is OFF

1 Insert your working copy of the MS-DOS System diskette into drive A and shut the drive door.

2 Turn on your computer.

A few seconds elapse as the system checks itself. The more memory in your computer, the longer this check-out period lasts. The following messages appear on your screen. The exact wording or sequence may vary depending on your computer configuration.

```
Resident Diagnostics
Rev 1.0 May 1984

CPU (i8086)          Pass
ROM Module           Pass
DMA Timer            Pass
DMA Control          Pass
Interrupts           Pass
128kb RAM            Pass
RT Clock             Pass
Fixed Disk           Not Present
Floppy (A:)          Ready

Primary Boot-Strap . . .

AT&T Personal Computer 6300
Copyright (c) 1984 by AT&T, all rights reserved

Compatibility Software Copyright (c) 1984
by Phoenix Software Associates, Ltd.

Microsoft MS-DOS version 2.11 release 2.0
Copyright 1981,82,83 Microsoft Corp.

Command v. 2.11

A>
A>
```

If Your Computer Is ON

1 Press the reset button.

2 Insert your working copy of the MS-DOS System diskette into drive A and shut the drive door.

The diskette drive light comes on while MS-DOS is being read into memory.

When MS-DOS Is Ready: The Prompt

After MS-DOS is loaded into memory, your screen looks like this:

```
Resident Diagnostics
Rev 1.0 May 1984

CPU (i8086)              Pass
ROM Module               Pass
DMA Timer                Pass
DMA Control              Pass
Interrupts               Pass
128kb RAM                Pass
RT Clock                 Pass
Fixed Disk               Not Present
Floppy (A:)              Ready

Primary Boot-Strap . . .

AT&T Personal Computer 6300
Copyright (c) 1984 by AT&T, all rights reserved

Compatibility Software Copyright (c) 1984
by Phoenix Software Associates, Ltd.

Microsoft MS-DOS version 2.11 release 2.0
Copyright 1981,82,83 Microsoft Corp.

Command v. 2.11

A>
A>
```

The A> on the screen is the MS-DOS prompt. Whenever this prompt appears, you know that MS-DOS is ready to go. A> prompts you, indicating that you must tell MS-DOS what to do next by entering a command.

Entering Commands

To make your computer perform a task, you must tell it what to do. This is called entering a command. The various MS-DOS commands perform different tasks such as displaying a list of files on a diskette (the DIR command), or making a duplicate of a diskette (the DISKCOPY command).

Most MS-DOS commands do one thing. You tell your computer to do "this" or "that" with a single command.

To give MS-DOS a command:

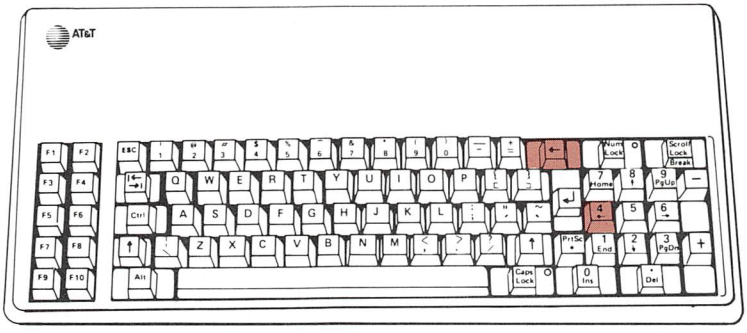
1. Wait until the A> appears.
2. Enter the command and any other information required. For example, it's sometimes necessary to include a drive specifier or a file specification in a command.
3. Press **RETURN** and the command procedure begins.

Entering an MS-DOS command is easy. Here are some simple rules:

- A command can be entered in uppercase or lowercase or any combination.
- Use a blank space to separate the parts of the command from one another.
- Always press the **RETURN** key after entering the command.

If You Make a Mistake

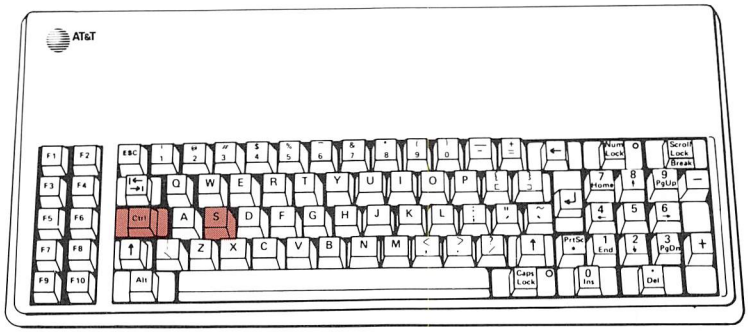
Sometimes you might make a mistake typing a command. You can correct an error two ways. You can press **BACKSPACE** which erases one letter at a time as it moves to the left, then retype your input. You can also use the **LEFT ARROW** to move the cursor to the left and make the necessary change.



Stopping the Screen to Read It

When the screen fills up with data, new information appears at the bottom, pushing what's at the top "up" and causing it to disappear from view. This process is called scrolling.

Some MS-DOS commands (for example, DIR and TYPE) cause text to be displayed on the screen at such a rapid rate that data may be scrolled out of sight before you get a chance to read it. In order to stop the display so that you can read what is on the screen, press and hold **CTRL** and then press the **S** key.



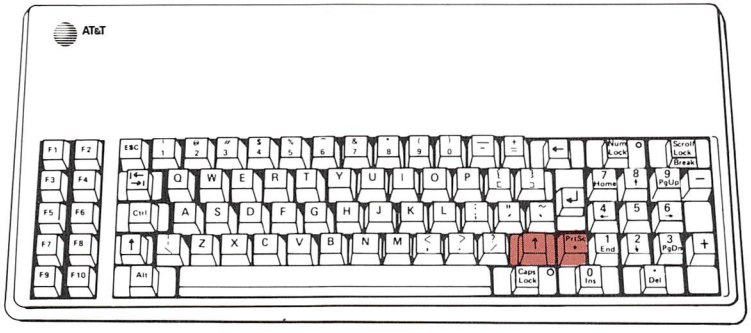
Press any key to resume the display of text.

Printing What Is on the Screen

You may want to have a printed copy of what is being displayed on the screen. For example, you may need a copy of a form or just one piece of information from a file.

If you are working with text or numbers and you have a printer attached to your computer, you can get an exact copy of what is displayed on the screen by pressing and holding **SHIFT** and then pressing **PRT SC**.

If you are working with graphics, you must have loaded the GRAPHICS Utility and must be sure that your printer is an AT&T 473 compatible graphics printer.



The Default Drive

The "A" in the prompt indicates the default drive. The default drive is the diskette or fixed disk drive that MS-DOS uses to perform the command you've entered. When you enter a command or filename, MS-DOS automatically searches the diskette located in the default drive for this information, unless you indicate another drive in the file specification. For example:

```
A>DIR
```

searches drive A for a directory of files. Since no drive is specified, the default drive A is assumed. However:

```
A>DIR  B:
```

searches drive B for a directory since drive B is specified.

It is possible to change the default drive in the prompt. Enter the new drive designation letter and follow with a colon. For example:

```
A>B:
```

When you press **RETURN** after this command, the new prompt appears:

```
B>
```

To find a directory on a diskette in drive A, you would now have to enter:

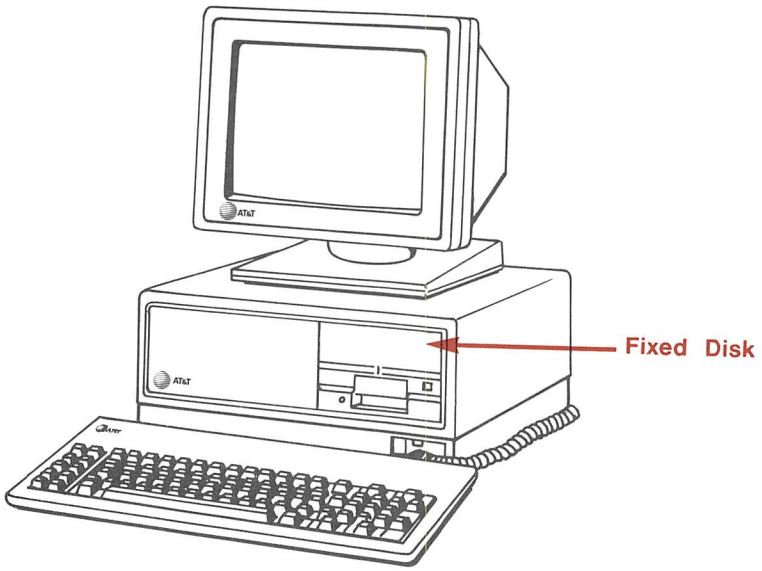
```
B>DIR  A:
```

Using a Fixed Disk Drive

If your computer is equipped with a fixed disk drive you must carefully consider how to best use your fixed disk before you prepare it to receive program and data files.

If MS-DOS is the only operating system you are ever going to use, then the procedure is simple. Refer to the section in this chapter called **Using MS-DOS Only** for a step-by-step guide.

If you are planning to use more than one operating system, or **may** choose to do so in the future, read through this entire section before starting to set up your fixed disk. If you are going to use more than one operating system you must allocate storage space on the fixed disk for each operating system. This is called "partitioning" the disk. Each operating system resides in its own partition.



Some considerations:

- How many people will be using this computer? They may want to have their own subdirectory and may want to use another operating system.

- You **can** partition the fixed disk for other operating systems later. However, you must perform several steps to remove existing files onto backup diskettes, then reformat and partition the fixed disk, before you can replace the existing files back into the MS-DOS partition.

- Despite the seemingly unlimited amount of storage available with a fixed disk, partitioning the disk for two or more operating systems divides the available space. Be certain that having more than one operating system is what you want.

- To set up other operating systems on your fixed disk drive, refer to the appropriate documentation for the other systems.

Using MS-DOS Only

If you are going to use MS-DOS exclusively with your fixed disk, follow these steps.

1 Start the computer with your working copy of the MS-DOS System diskette in drive A.

2 When the A> prompt appears, type:

FDISK

and press **RETURN**.

3 When the FDISK options menu appears choose #1 – Create MS-DOS Partition.

4 When you are asked "Do you wish to use the entire hard disk for MS-DOS (Y/N)?" Type

Y

and press **RETURN**.

The entire fixed disk is now allocated for MS-DOS operations.

When the FDISK options menu reappears, the following prompt will appear at the bottom of the screen:

Press **ESC** to return to **DOS**.

5 Press the **ESC** key.

The screen displays:

```
System will now reboot
Insert DOS diskette in drive A:
Press any key when ready . . .
```

6 Insert your MS-DOS System diskette in drive A and press any key.

The system restarts, and runs diagnostics. After a moment, the screen displays the A> prompt.

7 To format the fixed disk, wait for the A> prompt to appear, then type

FORMAT C:/S

and press **RETURN**.

This message is displayed:

```
!!!  WARNING  !!!
You are formatting the hard disk
THIS WILL DESTROY ALL DATA ON THE HARD DISK
Enter Y to continue, or any other key will cancel the
request to format.
Do you wish to format the hard disk?
```

8 Type **Y**

The screen displays:

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

9 Press a key to continue with the formatting process.

NOTE: Formatting a fixed disk takes several minutes. The screen displays the message "Formatting .....". When the process is finished, a message appears and the hidden system files are also placed on the fixed disk so you can start the computer from the fixed disk.

10 Copy all of the external MS-DOS commands and related programs to the fixed disk by typing:

```
COPY  *.*  C:
```

and press **RETURN**.

Your fixed disk can now be used to start your computer. Whenever you turn your computer ON or press the RESET button, or press and hold **ALT**, **CTRL** and then press **DEL** to restart the computer, leave drive A open and MS-DOS will select drive C as the default drive.

Partitioning Your Fixed Disk

If you want to use more than one operating system with your fixed disk, you must partition the disk. This allows each operating system to occupy space tailored to its exact needs for proper operation. If you are using an operating system other than MS-DOS, you may also want to partition a fixed disk if several people are using the same computer. A partition can be set up for each individual user or category of users.

When you are using MS-DOS, you can have only one MS-DOS partition.

Each operating system that can be used with a fixed disk has its own commands for placing it on the disk. Refer to that operating system's user's guide for instructions on placing it in its own partition on a fixed disk.

The MS-DOS program that performs the partitioning of your fixed disk is FDISK.COM. It is on the MS-DOS System diskette.

The FDISK command allows you to:

- Set the size of the MS-DOS partition
- Set a partition's position on the disk
- Select the active partition used when the computer is started
- Delete the MS-DOS partition
- Display the partition map.
- Perform the above operations on either of two fixed disks if you have added an extra fixed disk.

Creating the MS-DOS Partition

1 Type:

FDISK

and press **RETURN**.

In a few seconds, the FDISK menu appears on the screen.

```
Fixed Disk Setup Program

FDISK Options


Choose one of the following:

      1 Create MS-DOS Partition
      2 Change Active Partition
      3 Delete MS-DOS Partition
      4 Display Partition Information

Enter choice ..................................  [1]

Press Esc to return to DOS ....................  [ ]
```

If you have more than one fixed disk, the screen displays an extra option:

```
Select next fixed Disk Drive.
```

2 Choose #1 Create MS-DOS Partition from the FDISK menu.

3 If the disk has not been already set up you are asked if you want to use the entire fixed disk for MS-DOS. Type:

N

and press **RETURN**. This causes a message describing the total number of cylinders your fixed disk has and the size and location of the contiguous cylinders. If there is nothing on the disk, these two amounts are the same.

NOTE: Space on a fixed disk is measured in cylinders. On your computer a cylinder is approximately 34,000 characters of storage space.

4 If your fixed disk **is already set up** this is displayed:

```
Fixed Disk Setup Program

Create MS-DOS Partition

  Partition      Status      Type      Start      End      Size
      1            A         DOS         0       304      305

Total disk space is 305 cylinders.
The current active partition is 1

Disk already has an MS-DOS partition
Press Esc to return to FDISK Options. [ ]
```

This shows a typical partition map with three partitions on the fixed disk. This information describes

- the partition status, Active or Non-active
- whether or not the partition was allocated to MS-DOS
- the beginning and ending cylinder number of the partition
- the size of the partition
- total space on the fixed disk
- available space and location

5 Respond to the prompt

```
Enter partition size .....
```

by typing the number of cylinders you want to allocate for MS-DOS. Enter the number and press **RETURN**.

6 Respond to the prompt

```
Enter starting cylinder number ... nnn
```

by either pressing **RETURN** or entering another three-digit number. The default value displayed is the first cylinder of the smallest space that is large enough for the partition size you entered for step 5.

7 Respond to the prompt

Press **ESC** to return to the FDISK Option

by pressing **ESC.** You may want to check that the MS-DOS partition is active by going to the partition map display.

8 You should now restart your system by choosing the "Return to DOS" option. Be sure to have the MS-DOS System diskette in drive A.

9 Run FORMAT to format the MS-DOS partition on the fixed disk. Type:

```
FORMAT C: /S
```

10 When the format is complete, type:

```
COPY *.* C:
```

Steps 9 and 10 format the MS-DOS partition, move the hidden system files into the partition, then copy the MS-DOS external commands and related programs into the partition.

Your fixed disk is now ready to be used.

Changing the Active Partition

Enter the FDISK command and select #2 Change Active Partition from the menu. If you have more than two operating systems installed, be sure you know which one you want to make active.

Simply respond to the prompt

> Enter the number of the partition you want to make active .....

by typing the number of the partition you want to be the partition that starts your computer, and pressing **RETURN**. The displayed partition map is updated and the new partition is used as the startup file for your computer.

Deleting the MS-DOS Partition

Be careful. Deleting this partition destroys the contents of this part of the fixed disk. Be sure to make backup copies of the files. Use the MS-DOS command BACKUP before you continue.

To delete the MS-DOS partition, select #3 Delete MS-DOS partition from the FDISK menu. The partition map is displayed along with a "Warning" advisory.

To cancel the delete operation press **ESC**.

To proceed with the delete operation press **Y** and **RETURN**.

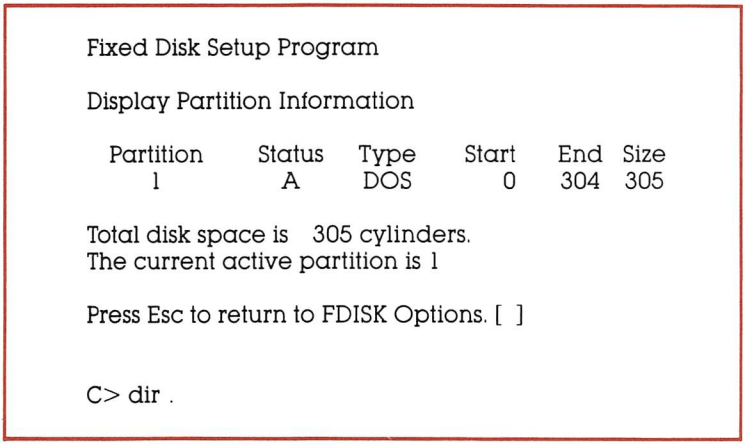
To restart your computer with MS-DOS after you have deleted the MS-DOS partition you must use an MS-DOS system diskette in drive A.

To restart your computer with another operating system you must either:

1. Select another active partition.
2. Use another system diskette in drive A.

Displaying the Partition Map

Option #4 Display Partition Data from the FDISK Option menu causes the partition map to appear on the screen.



```
Fixed Disk Setup Program

Display Partition Information

  Partition      Status    Type      Start     End   Size
      1            A       DOS         0      304   305

Total disk space is   305 cylinders.
The current active partition is 1

Press Esc to return to FDISK Options. [ ]


C> dir .
```

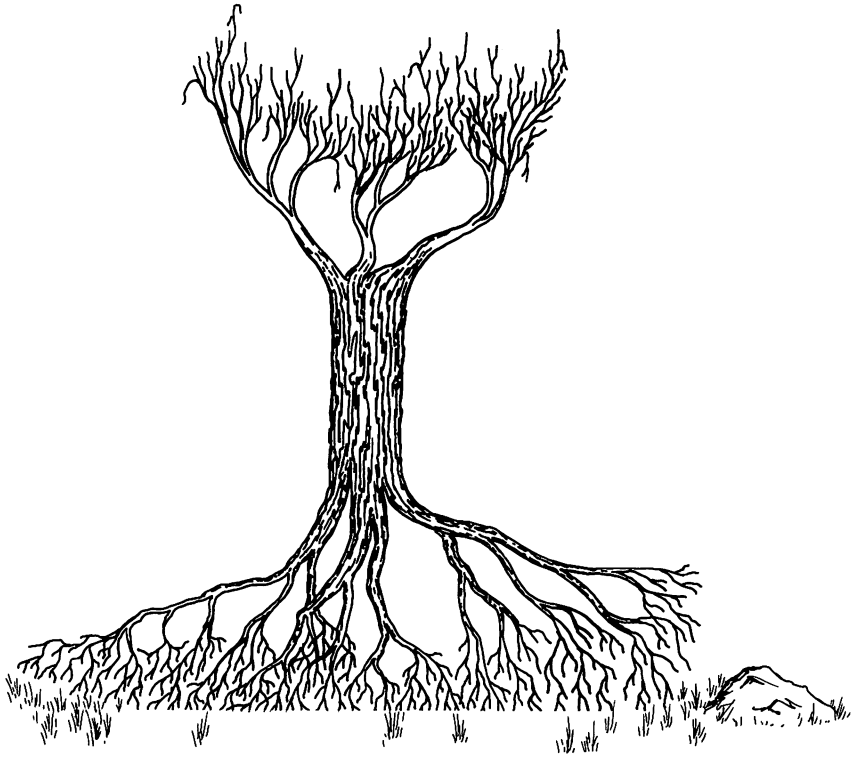
To return to the menu press **ESC**.

NOTE: Option #5 Select Other Disk will only appear if a second hard disk unit is installed.

Organizing Your Fixed Disk

The information contained in this section is "helpful hints" about using a fixed disk drive. For new computer users, or users who haven't used fixed disks before, putting all available disk storage to best use can pose a challenge.

To help you, MS-DOS has the capability to create subdirectories and sub-subdirectories. Imagine an upside-down maple tree. (The trunk is in the air and the main branches and limbs are balancing all of this lightly on the ground.)



When you issue the MS-DOS DIR command, a complete listing of all the files on the specified drive appears on the screen. What is displayed are all of the files on that disk or diskette. If you've been using your fixed disk for storage for all of your work and program files, this can be a very, very long listing.

The DIR command is showing you **everything** in the "trunk" or ROOT directory of the tree. If you have several hundred files on the disk — and this is not too difficult to do — it is both time consuming and tedious to locate a specific file if you've forgotten its exact name and extension.

What's more, if there are a large number of files in the directory, system performance slows down when your computer is looking for a specific file.

To solve these problems, you create "branches and limbs" (or subdirectories) of the main tree. By establishing a pathname the DIR command can follow, you speed up processing. With faster processing you get your own work done more quickly.

A Sample Multi-Level Directory

The following illustration shows how a fixed disk subdirectory structure might be set up.

Bill sets up his computer so that Mike and Sue, who work for Bill, have their own subdirectories. Mike is a financial analyst and frequently uses a spreadsheet program. Sue is a product manager involved in developing and testing new products.

Here is how Bill set up his disk directories.

Volume in drive C has no label
Directory of C:\

```
COMMAND     COM     15957     5-01-84     9:01a
AUTOEXEC    BAT         1     5-25-84     1:08p
CHKDSK      COM      6468     5-01-84     9:00a
COMP        COM      2766     5-01-84     9:00a
DISKCOPY    COM      2051     5-01-84     9:00a
FC          EXE      2585     5-01-84     9:00a
FDISK       COM      4464     5-01-84     9:00a
FORMAT      COM      5872     5-01-84     9:00a
GWBASIC     EXE     69008     5-01-84     9:01a
  .
  .
  .
BILL                <DIR>     7-01-84     8:30a
SPREAD              <DIR>     7-01-84     6:15p
WP                  <DIR>     7-02-84     9:45a
TEST                <DIR>     7-03-84     4:35p
```

In the main (ROOT) directory, Bill has placed all of the files of the MS-DOS operating system and related programs. Thus, all the MS-DOS functions are available to whomever is using the computer.

There are four subdirectories.

- A private directory for Bill
- The Spreadsheet directory that contains the spreadsheet program and two sub-subdirectories
- The Word Processing directory that contains the word processing program and two sub-subdirectories
- The Test subdirectory and two sub-subdirectories that contain files about the new products that Bill and Sue review.

Bill mapped out the fixed disk directories before he assigned them names. Then he used the MD (Make Directory) and CD (Change Directory) commands to create this structure. Here's how he did it:

1 Loaded MS-DOS into memory.

2 Created his personal directory by typing:

```
MD BILL
```

This created the subdirectory BILL under the ROOT directory. Bill keeps his own programs and special files here.

After he starts his computer and the MS-DOS prompt appears, he types

CD BILL

to get into his own directory.

3 To create the SPREAD subdirectory for the spreadsheet program and **its** subdirectories Bill typed:

MD C:\SPREAD

First, he created the subdirectory SPREAD under the ROOT directory. Then he changed directory (CD) to SPREAD by typing

CD SPREAD

Next, Bill typed:

MD BILL

and

MD MIKE

to create a subdirectory for himself and for Mike. The MD command can also be entered as MKDIR.

4 Bill repeated this process for the word processing directory (WP) with the commands:

CD (return to ROOT directory)

Then he typed:

```
MD C:\WP
```

to make a directory on drive C: named "WP". He then used the Change Directory command to enter the new WP directory by typing:

```
CD WP
```

and created two subdirectories for his people who would be using the word processing directory, WP by typing in the commands:

```
MD BILL
```

and

```
MD SUE
```

5 To complete the directory-building process Bill typed in the following commands to create subdirectories under the directory named TEST:

```
CD \
MD C:\TEST
CD TEST
MD BILL
MD SUE
```

Using Subdirectories

Whenever Sue wanted to get into the files she was keeping on the new products undergoing testing, she started the computer and issued the following commands:

```
CD \TEST\SUE
```

which identifies the path to her personal subdirectory under the TEST directory, and

```
DIR
```

which displays a directory listing of all her files under that subdirectory so that she could select the file she wanted to work with.

By dividing up each user's files into subdirectories Bill provided an easy way for each computer user to access his or her own files out of the 200-odd files that were stored on the fixed disk. In this way, operation of the computer's file storage disk was speeded up, and making backup copies of each person's work files was made easier.

4 Using MS-DOS Commands

- MS-DOS Commands for Common Tasks
- Formatting a Diskette
- Finding Out What Files Are on a Diskette
- Copying a Diskette
- Comparing Diskettes
- Copying a File
- Comparing Files
- Looking at a File
- Changing a File Name
- Removing a File from a Diskette
- Processing a Series of Commands Automatically
- Helpful Hints

MS-DOS Commands for Common Tasks

In this chapter, you will learn about the most frequently used MS-DOS commands, what you need to carry out each of them, and how to proceed step-by-step. These MS-DOS commands are the commands that you are going to use so often that they become second nature.

What you learn in this chapter will help you on your own to learn to use all MS-DOS commands.

In the process of starting MS-DOS, some commands are loaded into memory where they stay until you need them. These are the internal commands. You don't need a copy of the MS-DOS System diskette in a drive in order to use one of them. The common internal commands and their functions are:

Command:	**Function:**
COPY	Copying a file
DIR	Finding out what files are on a diskette
TYPE	Looking at a file's contents
RENAME	Changing a file's name
DEL	Removing a file

Most MS-DOS commands require that you have a copy of the MS-DOS System diskette in a drive. These are called external commands. The most common external commands and their functions are:

Command:	Function:
FORMAT	Getting a diskette or fixed disk ready for use
DISKCOPY	Copying diskettes
DISKCOMP	Comparing diskettes
COMP	Comparing files

Formatting a Diskette

The FORMAT command prepares a diskette to receive information. You only need to FORMAT a diskette once, when it is new. FORMAT sets up the "tracks" and "sectors" where data is written on the diskette or fixed disk by your computer. FORMAT also checks the diskette or fixed disk to make sure it has no flaws, then creates a directory to find the data you eventually write on it.

Be careful. FORMAT removes all of the information on a diskette or fixed disk. Be sure the diskette you are going to FORMAT does not contain information you want to keep.

What Is Needed

- the MS-DOS System diskette
- the diskette you want to format

Procedure

If you have a single diskette drive, MS-DOS instructs you to switch diskettes. Do not remove or replace a diskette until you are told to do so.

1 Load MS-DOS and wait until A> appears.

2 Leave the MS-DOS System diskette in drive A.

3 Type:

```
A>format b:
```

and press the **RETURN** key. The drive in your AT&T PC is dual-sided, and this creates a diskette for dual-sided use only.

This message appears on the screen:

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

4 FOR TWO DRIVES: Insert the new diskette into drive B and press any key.

WITH ONE DRIVE: When the drive A light goes off, remove the MS-DOS System diskette from drive A, insert the new diskette into drive A, and press any key.

5 While the diskette is being formatted,

```
Formatting ...
```

appears on the screen. After the procedure is finished, you see:

```
Formatting ... Format complete

  362496 bytes total disk space
  362496 bytes available on disk

Format another (Y/N)?
```

NOTE: The message on your screen may also show some number of bytes in bad sectors. If the screen message shows bad sectors, try formatting again. Discard the diskette if there are still more than 3000 bytes in bad sectors.

6 If you have more diskettes to format, type **Y** for yes and repeat the above steps.

To stop formatting, type **N** for no. The MS-DOS prompt appears and you can remove the newly formatted diskette.

If you include **/S** after the command FORMAT, MS-DOS copies the system files to the target diskette as well as formatting the diskette. This type of FORMAT command is entered as:

```
A>format  b: /s
```

The result is a disk that starts MS-DOS directly. If you add system files to a formatted diskette, the screen shows:

```
362496 bytes total disk space
 43008 bytes used by system
319488 bytes available on disk
```

NOTE: A disk with bad sectors on track 0 cannot be used as a system disk since track 0 is where the information that boots the computer must be stored. It can still be useful as a data disk. If there are too many bad sectors, the disk should be discarded.

Finding Out What Files Are on a Diskette

The DIR command displays the names and file statistics of the files on a diskette. When you are looking for a specific file and you don't know on which of your diskettes it is located, the internal DIR command helps you find what you are looking for.

What Is Needed

- the diskette whose file directory you want to check

If MS-DOS is already in memory, you don't need the MS-DOS System diskette for this procedure.

Procedure

1. Make sure MS-DOS is ready and A> is displayed.
2. Insert the diskette you want to check into drive A.
3. Type the command:

   A>**dir**

   and press the **RETURN** key.

4 The following is displayed on the screen:

- the volume label of the diskette (if it has one)
- the name of the directory that is being listed
- the directory's files — one line is displayed for each file with this information:
  - filename
  - extension
  - size (in bytes)
  - date and time that information was last written in the file
- the amount of free space left on the diskette (in bytes)

5 When all the files have been displayed, A> appears.

NOTES: If the screen is scrolling too fast for you to read the files, use the **CTRL** and **S** keys to stop it.

To print what appears on the screen, use the **SHIFT** and **PRT SC** keys.

If you type A> **dir /P** and press the **RETURN** key, the directory display halts as soon as the screen is full (/P means Page). Press any key to resume the listing.

If you type A> **dir /W** and press the **RETURN** key, only the filenames are displayed, five to a line across the screen (/W means wide). This lets you look at a large directory at a glance.

Copying a Diskette

ALWAYS make duplicates of your important diskettes.

Why You Should Always Make Copies of Diskettes

This is a rule that every computer user learns the hard way. To prevent the bother of having to re-do hours of work or replace vital programs, make copies of your files and put them away for a rainy day.

How to Copy Diskettes

The DISKCOPY command makes a duplicate of an entire diskette on another diskette. As you learned in Chapter 1, it's a good idea to make working copies of your program diskettes as soon as you get them. Use the copies for daily work.

NOTES: The date and time shown with a directory entry are the date and time of the last addition or change to that file. The date and time are not changed when you use the DISKCOPY command.

Use the internal command COPY to move files between a fixed disk and a diskette, or to copy only specific files from one diskette to another.

What Is Needed

- the MS-DOS System diskette
- the original — or source — diskette
- the blank — or target — diskette that's going to become the backup copy

NOTES: If the target diskette you use is not blank, but has files on it, all of these old files are wiped out during the procedure. That's because DISKCOPY makes an exact duplicate of the original diskette onto the target diskette.

DISKCOPY will also format the target diskette if it has not already been formatted.

Procedure

If you have one diskette drive, MS-DOS treats it as both drive A and B. Switch diskettes each time MS-DOS tells you to exchange diskettes.

1 Load MS-DOS and wait for A> to appear.

2 Insert the MS-DOS System diskette into drive A.

3 Type:

A>**diskcopy a: b:**

and press the **RETURN** key.

4 WITH TWO DRIVES: This message appears:

Insert source diskette in drive A:
Insert target diskette in drive B:
Strike any key when ready

Remove the MS-DOS System diskette, insert the source diskette into drive A and the target diskette into drive B. Press any key when ready.

If the target disk is not formatted, or is formatted differently from the source disk, DISKCOPY will display:

Formatting while copying

and will format the target disk as well as copy the files to it, so that it is an exact duplicate of the source disk.

WITH ONE DRIVE: This message appears:

Insert source diskette in drive A:
Strike any key when ready

Remove the MS-DOS System diskette that is in drive A. Insert your source diskette into drive A and press any key.

5 WITH TWO DRIVES: You'll notice that the lights on drives A and B go on and off as the copying proceeds. Do not open either drive door at any time during the process. DISKCOPY is complete when the following appears:

```
Copying complete

Copy another (Y/N)?
```

WITH ONE DRIVE: The in-use light comes on while the source diskette is being read. When the computer has read the entire source diskette (or as much of it as can fit in memory), this is displayed:

```
Insert target diskette in drive A:
Strike any key when ready
```

Remove the source diskette, insert the target diskette, and press any key. The light is on while the copy is being written. Depending on the amount of memory in your computer, the request to insert the source diskette may reappear. You may be required to repeat this diskette exchange process several times until you see:

```
Copying complete

Copy another (Y/N)?
```

6 If you have more diskettes to copy, type **Y** for yes and follow the steps again.

If you are finished, type **N** for no.

When the A> appears, remove the target diskette, prepare an appropriate label, apply the label to the diskette and store it in its paper sleeve.

Comparing Diskettes

The DISKCOMP command makes sure that the duplicate diskette you've just made was copied accurately. In most cases the DISKCOPY command discovers any problems while it is performing the copy, but sometimes there is no warning that something has not worked out just right. The DISKCOMP command compares the source and destination disks to make sure they are exact duplicates. It is a good idea, particularly with important data files or valuable programs, to double check by using the DISKCOMP command.

What Is Needed

- the MS-DOS System diskette
- the original diskette
- the new duplicate diskette

Procedure

If you have one drive, MS-DOS treats it as both drive A and B. Switch diskettes each time MS-DOS tells you to exchange diskettes.

For this procedure, the source or original diskette is the first diskette, and the destination or duplicate is the second diskette.

1 Load MS-DOS and wait for A> to appear.

2 Insert the MS-DOS System diskette into drive A.

3 Type:

```
A>diskcomp a: b:
```

and press the **RETURN** key.

4 WITH TWO DRIVES: This message appears:

```
Insert first diskette in drive A:
Insert second diskette in drive B:
Strike any key when ready
```

Insert the first diskette into drive A, the second diskette into drive B, and press any key.

WITH ONE DRIVE: This message appears:

```
Insert first diskette in drive A:
Strike any key when ready
```

Remove the MS-DOS System diskette from drive A, insert the source (original) diskette into drive A, and press any key.

5 WITH TWO DRIVES: The drive lights go on and off as the computer compares the two diskettes. Do not open the drive doors during this process. When the process is complete, the screen looks like this:

```
A> diskcomp a: b:
Insert first diskette in drive A:
Insert second diskette in drive B:

Strike any key when ready

Comparing 9 sectors per track, 2 sides

Diskettes compare ok

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

WITH ONE DRIVE: The drive light comes on while the source diskette is being read, and after a few moments, this message is displayed:

```
Insert second diskette in drive A:
Strike any key when ready
```

Remove the source diskette from drive A, insert the target (duplicate) diskette into drive A, and press any key.

The light comes on once again while the duplicate diskette is being compared.

Depending on the amount of memory in your computer, you may have to switch the diskettes several times until this message appears:

```
Diskettes compare ok

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

6 If you have more diskettes to compare, type **Y** for yes and repeat the procedure.

If you don't want to compare any more diskettes, type **N** for no.

You don't have to press the **RETURN** key. After the MS-DOS prompt appears you can remove the diskette(s) from the drive(s).

If the diskettes are not identical, go through the copying and comparing procedures again. If the diskettes do not compare successfully, use another target diskette. The first target diskette is probably bad and should be re-formatted. Format the troublesome diskette, then repeat the copy and compare procedures once again.

If this last attempt does not solve the problem, discard the bad diskette and use another one.

Copying a File

Use the COPY command when you want to copy only a few files instead of an entire diskette, or when you are copying to or from a fixed disk.

Sometimes copying a single file to a target diskette is a good way of archiving your work. You might want to keep all of your copies of specific types of files on a special archive diskette. All the company memos are copied to the "Company Memos" diskette, your travel expense records are copied to the "T&E" diskette, etc. In another case, database files on a fixed disk should be copied onto archival diskettes after every update of the information. The internal COPY command is the one to use.

NOTE: The date and time shown with each directory entry are not changed during a COPY.

What Is Needed

- MS-DOS loaded into your computer's memory
- the diskette that contains the file you want to copy (the source diskette)
- a formatted diskette that is to receive the copy of the file (the target diskette)

Be careful. If files already exist on the target diskette, COPY does not disturb these old files as long as their names aren't the same as those of files being copied. If you do copy a new file onto a diskette with an old file of the same name, the old file will be replaced.

If you COPY onto a diskette with other files, make sure in advance there is enough space (in bytes) left to hold the new file(s).

NOTES: The COPY command also lets you merge two or more files into a single target file. Refer to Chapter 5 for more information about merging files with the COPY command.

Use the COPY command with two wild cards (COPY A:*.* B:) to move all the files from a well-used diskette to a fresh one. COPY places each file on the target disk one after the other. The source disk can then be erased to recover space on the disk to receive new files.

Procedure

If you have one diskette drive, MS-DOS treats it as both drive A and drive B. Switch diskettes each time MS-DOS tells you to exchange diskettes.

1 Load MS-DOS and wait for the A> to appear.

2 Remove the MS-DOS System diskette and insert the source diskette into drive A.

3 WITH TWO DRIVES: Insert the destination diskette in drive B. Type:

A>**copy filename b:newfilename**

and press the **RETURN** key.

NOTES: The filename of the copy can be the same as the original or you can give it a new name.

Make sure you put spaces where indicated in the command.

Wait until the screen message

```
1 file(s) copied

A>
```

appears.

4 WITH ONE DRIVE: Enter the command:

A>**copy filename b:newfilename**

and press the **RETURN** key.

This message appears:

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

Remove the source diskette from the drive, insert a formatted target diskette, and press any key.

Depending on the size of your computer's memory you may have to switch the diskettes several times to complete the procedure.

5 When the copy has been made, the following appears:

```
1 File(s) copied

A>
```

Remove the target diskette from the drive, label it with a felt-tip pen, and store both diskettes in paper sleeves in a safe place.

Comparing Files

The COMP command lets you know whether the file or group of files you copied using the COPY command is identical to the source file(s).

NOTE: COMP cannot be used to check files you have stored on the diskette using the BACKUP command.

It is a good idea to check to make sure your copy of a file is the same as the original, particularly if that file is important. You can compare files immediately after the COPY operation or any time thereafter.

COMP differs from DISKCOMP in that COMP works on individual files rather than complete diskettes. COMP is also faster than DISKCOMP.

What Is Needed

- the MS-DOS System diskette
- the diskette containing the original (source) file(s)
- the diskette containing the copied (target) file(s)

Procedure

If you have one diskette drive, MS-DOS treats it as both drive A and drive B. Switch diskettes each time MS-DOS tells you to exchange diskettes.

1 Load MS-DOS and wait for A> to appear.

2 Insert the MS-DOS System diskette in drive A if it isn't there already.

3 Type:

```
A>comp
```

and press the **RETURN** key.

When you get the message:

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

insert the diskette to be compared and press any key.

4 When this message appears:

Enter primary file specification

enter the filename from the source diskette, like this:

a:filename

Press the **RETURN** key.

5 The next message to appear is:

Enter secondary file specification

Enter the name of the file you copied to the target diskette:

b:filename

6 WITH TWO DRIVES: Insert the target diskette into drive B (the original should already be in drive A) and press the **RETURN** key. The names of the files being compared are displayed.

WITH ONE DRIVE: With the source diskette in drive A, press the **RETURN** key. Shortly, this message appears:

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

Remove the source diskette from drive A, insert the target diskette into drive B, and press any key.

If the size of the files being compared is very large, you may have to switch diskettes several times and repeat steps 3 and 6. The files are placed in memory and some files may be larger than the portion of memory allocated during the compare process.

7 When the files have been compared and found to be identical, this message appears:

```
Files compare ok

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

8 If you have more files to compare, type **Y** for yes and repeat the procedure.

If you are finished, type **N** for no.

The MS-DOS prompt is displayed and you can go on with the next operation.

If the files are not identical, a number of advisory messages appear on the screen, and the COMP program stops. Go through the copying and comparing steps again.

Looking at a File

The TYPE command lets you display the contents of a file on your screen. You may want to do this when you are uncertain about which file you want to do some work on. The internal TYPE command lets you quickly review the text of a file to check if it is the one you want. This way, you don't have to start another program (for example, a word processor) in order to check the contents of a file.

If you use the TYPE command with a program file, the results displayed on your screen may not be understandable. Because program files contain control characters or are in a format best used by a computer rather than read by a person, the contents of these files often are best examined by using the programming language with which they were created. Experimenting with program files and the TYPE command does not harm the program file, but you may need to RESET your computer to continue.

What Is Needed

- the diskette with the file you want to display

If MS-DOS is already loaded into your computer's memory, you don't need your MS-DOS System diskette for this procedure.

Procedure

1. Make sure MS-DOS is ready and wait for A> to appear.

2. Insert the diskette into drive A.

3. Enter the name of your file with the command:

   A>**type yourfile**

   Press the **RETURN** key.

4. The command just entered and the file contents are now displayed.

   To stop the scrolling hold **CTRL** and press the **S** key. Press any key to resume scrolling.

5. When A> reappears, you can remove the diskette.

Changing a Filename

The RENAME command lets you change a filename — its name, extension, or both. You may want to change a filename to more closely identify the contents of a file that started out to be one thing, but ended up another. Other times, the RENAME command can be used to identify groups of files with common attributes.

Another way of renaming a file is to use the COPY command. The command syntax is the same as RENAME, the difference being that you then have two copies of a file, one with the original name on one diskette, and another with the new name on another (or the same) diskette or fixed disk drive.

What Is Needed

- the diskette with the file you want to rename

If MS-DOS is already loaded, you don't need the MS-DOS System diskette for this procedure.

Procedure

1 Make sure MS-DOS is ready and the A> appears.

2 Insert the diskette into drive A, if it's not already there.

3 Enter the command, plus the name of the old and new filename:

A>**rename oldfilename newfilename**

Press the **RETURN** key.

NOTE: The current filename is first, followed by a space, and then the new filename.

4 Once the operation is finished, the MS-DOS prompt reappears. Remove the diskette from the drive and replace it in the paper sleeve.

Removing a File from a Diskette

Be careful with this command. Once a file has been ERASEd, it is gone forever from that diskette or fixed disk drive.

The ERASE command and the DELete command are identical. You can use one or the other and the file or files specified are removed from the directory. Make sure that ERASEing a file is what you want to do.

Be very careful about using the ERASE or DELete commands when using wild card characters.

It is a good idea to "clean up" your diskettes from time to time. Remove obsolete or duplicate files to make more room for new files. Also, if a diskette has had a lot of activity — files added and removed — the amount of space on the diskette may be broken into a number of 'empty' spots which are inefficient for saving new files. See the COPY command for a tip on recovering small spaces on a diskette.

NOTE: Use this command with great care. Always double check your typed entries before pressing the **RETURN** key. Once ERASEd, a file cannot be retrieved.

What Is Needed

- the diskette with the file you want to erase

If MS-DOS is already in your computer's memory, you don't need the MS-DOS System diskette for this procedure.

Procedure

1 Make sure MS-DOS is ready and the A> prompt appears on the screen.

2 Insert the diskette with the file you plan to erase into drive A.

3 Enter the command, plus the name of the file to be deleted:

```
A>erase filename
```

Check your typed entry.

Press the **RETURN** key.

4 Shortly, the A> reappears. Remove the diskette from the drive, relabel it, and return it to its paper sleeve.

Processing a Series of Commands Automatically

MS-DOS allows you to create a file containing a series of MS-DOS commands that are immediately performed, one after another. This is called batch processing and the files used are called batch files.

You could use a batch file to start other programs on a diskette or to perform some routine startup tasks that you usually do whenever you turn on your computer. If you want your computer to do some regular task each time you turn it ON or whenever you restart it with the **ALT CTRL** and **DEL** keys, you create a file called AUTOEXEC.BAT.

The AUTOEXEC.BAT File

Whenever you power ON or restart your computer, MS-DOS looks on the startup drive for a file named AUTOEXEC.BAT. This file must be on the default drive. If you are using a two-diskette system, the file must be in drive A. If you have a computer with a fixed disk drive, MS-DOS looks on the fixed disk for AUTOEXEC.BAT. When found, MS-DOS automatically performs each of the commands in the file.

For example, you can redirect information being sent to the default printer port (LPT1:) over to a serial communications port (COM1:), then run a GWBASIC program called INVOICING with the following AUTOEXEC.BAT file:

```
MODE COM1:12,N,8,1,P
MODE LPT1:=COM1
GWBASIC INVOICING.BAS
```

The first line invokes the MS-DOS MODE command and defines the setup parameters for the serial printer attached to the serial port COM1.

The second line uses the MODE command to redirect output from the parallel printer port LPT1 to COM1.

The third line loads GWBASIC from the default diskette then loads the program called INVOICING.

NOTES: When you use an AUTOEXEC.BAT file, MS-DOS does not prompt you for the date and time entries unless you include the DATE and TIME commands in the file.

An AUTOEXEC.BAT file may contain either one command or a long series of commands.

See Chapter 6, **Batch Processing Commands,** for more information.

Helpful Hints

- Make copies of your important program and data diskettes regularly. Always date the label.
- Print a directory frequently and store the listing with the diskette.
- If a command does not work as it should:

  1. Check your typing.
  2. Make sure you have the correct diskette in the correct drive.
  3. Check the directory of the diskette with the DIR command.
  4. Make sure colons and spaces have been included where they should be and not where they don't belong.
  5. Make sure the filename is correctly spelled and includes the extension where appropriate.

- If a command still does not work the way you expect it to, refer to Chapter 5, **MS-DOS Commands,** for more information about that command.

5 MS-DOS Commands

This chapter describes the MS-DOS commands that are not specifically dedicated to batch processing files. The commands are in alphabetical order.

This chapter covers:

- **Command Syntax**
- **MS-DOS Commands**

Command Syntax

The following notation indicates how you should enter MS-DOS commands. Just like the rules of English grammar and usage, MS-DOS commands must be entered in an exact form — called command syntax — in order to have the correct action happen.

The following rules of command entry are used throughout this chapter to explain the proper way to express an MS-DOS command and all of its optional modifiers.

- Words that are printed in CAPITAL LETTERS are words that you must enter, and you must enter them exactly as they are spelled and punctuated. You may type them in either upper case or lower case, however.

- Anything enclosed in angle brackets < > must be entered. You supply the text. For example, enter the name of your file when <filename> is asked for in a command. Don't type the angle brackets. This would cause a problem.

- Items in square brackets [ ] are optional. Enter only the information asked for within the brackets.

  Don't type the square brackets. This would cause a problem.

- Items in braces { } indicate that you must make a choice between two entries. You must enter one or the other unless those choices are enclosed in surrounding square brackets.

- An ellipsis (...) indicates that you may repeat an entry as many times as you need.

- Enter all punctuation shown (with the exception of angle and square brackets, ellipses, braces, or in special cases, vertical bars), such as commas, equal signs, question marks, colons, or slashes. These punctuation marks are essential and must be entered exactly as shown.

- A vertical bar (|) appearing in a command entry means one of two things. In one case, the | indicates a choice between two required entries. It means "this" or "that" choice must be made. See the BREAK command for an example. In this case, the vertical bar is not typed in the command line.

  The ¦ symbol is used in the command entry to pass the results from one command to the next command on the command line. The results from the first command are passed to the second MS-DOS command for further processing. The second MS-DOS command is a "filter." The ¦ symbol is called a "pipe."

See the SORT command for an example of piping the output from one command to another.

DIR ¦ SORT > DIREC.FIL

sends the results of a DIR command to the SORT command, then writes the SORT results into a file named DIREC.FIL. Note the use of the > symbol.

MS-DOS normally assumes that input comes from your keyboard and that output is displayed on your screen. You can redirect the flow of input and output using the characters > and <. Input can come from a file instead of the keyboard, and output can go to a file instead of the screen. The pipe character can also be used with redirection characters to make complex MS-DOS operations easier to do.

For example,

DIR

displays a directory listing on your screen.

If you enter

DIR >DIRFILE

the output of the DIR command is written into a file on the default drive named DIRFILE. If the file named DIRFILE is not already on the default drive, it is created. If it already exists, the output from the DIR command writes over the contents of the existing file.

To **append** the output of the DIR command to the existing DIRFILE, use two >> symbols. For example,

DIR >> DIRFILE

appends the output from DIR to an existing file. If DIRFILE doesn't already exist, it is created. See the SORT command for an example.

MS-DOS Commands

The following MS-DOS commands are described in this chapter.

Note that synonyms for commands are enclosed in parentheses.

ASSIGN	Routes requests for one drive to another
BACKUP	Backs up files from a fixed disk
BREAK	Sets CONTROL-C check
CHDIR	Changes directories; prints working directory (CD)
CHKDSK	Scans the directory of the default or designated drive and checks for consistency
CHMOD	Changes/displays file and/or directory attributes
CLS	Clears screen
COMP	Compares files
COPY	Copies file(s) specified
CTTY	Changes the device used by MS-DOS for command input
DATE	Displays and sets date
DEL	Deletes file(s) specified (ERASE)
DIR	Lists requested directory entries
DISKCOMP	Compares the contents of two diskettes
DISKCOPY	Copies diskettes
ERASE	Same as DELete
EXE2BIN	Converts executable files to binary format

FC	Compares files
FDISK	Sets fixed disk partitions
FIND	Searches for a constant string of text
FORMAT	Formats a disk to receive MS-DOS files
GRAPHICS	Enables graphics from the screen to be printed on a graphics printer
MKDIR	Makes a directory (MD)
MODE	Sets display, communications, and serial printer environments
MORE	Displays output one screen at a time
PATH	Sets a command search pathname
PRINT	Background print feature
PROMPT	Designates command prompt
RECOVER	Recovers a bad disk
RENAME	Renames first file as second file (REN)

RESTORE	Restores files saved with BACKUP
RMDIR	Removes a directory (RD)
SET	Assigns a value to an environment variable
SHIP	Prepares hard disk for shipping
SIZE	Lists filenames and file sizes of specified files
SORT	Sorts data alphabetically, forward or backward
SYS	Transfers MS-DOS system files from one drive to another
TIME	Displays and sets time
TREE	Displays directories and their contents
TYPE	Displays the contents of file specified
VER	Prints MS-DOS version number
VERIFY	Verifies writes to disk
VOL	Prints volume identification number

ANSI.SYS

Purpose

ANSI.SYS Escape sequences are series of characters that you can use to define functions to MS-DOS. For information about ANSI.SYS Escape sequences, see **Appendix G.**

Purpose

The ASSIGN command tells MS-DOS to direct all requests from one drive to another drive.

Syntax

ASSIGN [drivespec1 = drivespec2]

Comments

Enter only the drive letters. Do not enter the colons on the command line.

Drivespec1 is source drive, drivespec2 is the new destination drive.

Entering the ASSIGN command without any original or destination parameters resets the current assignments to the default values.

ASSIGN could be useful in an AUTOEXEC.BAT file.

ASSIGN works unpredictably when used with the PRINT command. Do not use in this combination.

DISKCOPY and DISKCOMP commands ignore drive assignments.

BACKUP

Purpose

The BACKUP command makes backup copies of one or more files from a fixed disk onto a diskette(s).

Syntax

BACKUP <d:> [pathname] [filename]
<d>: [/S] [/A] [/M|/D:mm-dd-yy]

Comments

Use this command when you want to make copies of files for archival purposes, to recover unused space on your fixed disk drive or you are going to create new partitions on the fixed disk.

The first parameter you enter is the fixed disk file(s) to back up.

The second parameter is the backup disk drive.

Be careful. Unless otherwise specified, the copies of the old files on the backup diskette are erased before the new files are written on it.

This backup program is compatible with those supplied by most other manufacturers.

The BACKUP command sets exit codes that can be used in a batch processing file with an IF command. These exit codes are:

0	normal completion
1	no files found
2	BACKUP stopped by user
3	BACKUP stopped by error

Files duplicated with the BACKUP command can only be used by the RESTORE command to restore a fixed disk drive's files. Do not use BACKUP files for archival purposes.

Options

The following options may be used with the BACKUP command:

/S Back up subdirectories also.

/M Only back up those files that have changed since the last backup.

/A Add the files to be backed up to those already on the working copy. This does not erase old files on the diskette.

/D Only back up those files that were last modified at or after a certain date. The date must be entered in the format mm-dd-yy.

Purpose

The BREAK command is an MS-DOS command that sets the CONTROL-C check.

Syntax

BREAK [ON|OFF]

Comments

You use this option to prevent MS-DOS from being affected by a **CTRL C** entry when the program you are using uses these keystrokes for another purpose.

Select BREAK OFF to turn off CONTROL-C when you are running a program that uses these keystrokes. Select BREAK ON when you return to MS-DOS.

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

Set BREAK ON to stop assemble or compile processes. See the *System Programmer's Guide* for more information.

CHDIR
(CHange DIRectory)

Purpose

The CHDIR command changes the current directory to another pathname or it displays the current subdirectory you are using.

Synonym

CD

Syntax

CHDIR [pathname]

Comments

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

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

and MS-DOS puts you in the new directory.

There is a simpler notation you can use with this command:

CHDIR ..

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

If you enter CHDIR without a pathname, your working directory is displayed. For example, if your working directory is \BIN\USER\JOE on drive B, and you type

CHDIR

and press the **RETURN** key, the screen displays:

B:\BIN\USER\JOE

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

To return to your ROOT directory, type either

CHDIR

or

CD

CHKDSK
(CHecK DiSK)

Purpose

The CHKDSK command scans the directory of a specified disk to check that it is consistent with the files on the disk.

Syntax

CHKDSK [d:] [filename] [/F] [/V]

Comments

It is a good idea to use CHKDSK from time to time to check for errors in a diskette or fixed disk directory. If errors are found, CHKDSK may display error messages.

CHKDSK displays a status report.

This is a sample status report:

```
160256 bytes total disk space
  8192 bytes in 2 hidden files
   512 bytes in 2 directories
 30720 bytes in 8 user files
121344 bytes available on disk

 65536 bytes total memory
 53152 bytes free
```

Options

The following options may be used with the CHKDSK command:

/F Fix errors found in the directory.

CHKDSK may display one of the following error messages.

- Invalid drive specification
- Invalid parameter
- Invalid subdirectory entry
- Cannot CHDIR to ROOT
  Processing cannot continue
- First cluster number is invalid
  entry truncated
- Allocation error, size adjusted
- Has invalid cluster, file truncated
- Disk error reading FAT
- Disk error writing FAT
- <filename> contains non-contiguous blocks

Call your dealer for assistance if you cannot fix any of these problems.

/V Display messages on the screen while CHKDSK operates.

You can redirect the output from CHKDSK to a file. Simply type:

CHKDSK A:>filename

The errors are sent to the filename specified. Don't use the /F option if you send the CHKDSK error messages to a file.

Purpose

Change and display file and/or directory attributes

Syntax

CHMOD <d:>[pathname][filename]
[+|-S][+|-R][+|-H][+|-A][+|-V]

Comments

CHMOD is used to change, modify or list file or subdirectory attributes. The file attributes can be combinations of the following:

S System file — files used by the operating system. Normally these files are "hidden", in that they do not appear in the display from the DIR command.

R Read-only file — these files can only be read from, not written to.

H Hidden file — another file hidden from normal use by DOS.

A Archive file — a file that has not been backed up. Every time DOS creates or modifies a file this attribute is turned on.

V Volume ID — not a file. An 11 character ID name to help in the identification of diskettes.

You can set or clear any of the file attributes except the Volume ID and the DIRECTORY attributes. Global characters are allowed in filenames. You may enter one or more attributes at a time.

To set an attribute for any file, enter a plus (+) sign after the filename followed by the attribute letter-code.

If you enter a letter without the plus (+) or minus (−) sign, the corresponding attribute is set.

To clear an attribute, enter a minus (−) sign followed by the attribute letter-code to be cleared.

Examples

```
CHMOD A:*.*
```

displays attributes for the current directory on drive A.

```
CHMOD B:\wp\*.dat
```

displays attributes for all filenames with the extension .DAT in the subdirectory wp.

```
CHMOD diskcopy.exe +h+r
```

sets the HIDDEN and READ ONLY attributes for the filename DISKCOPY.EXE.

```
CHMOD b:\overlays\wsmsgs.ovr -r
```

clears the READ ONLY attribute from the filename MSMSGS.OVR in the subdirectory OVERLAYS on drive B.

Entering

```
CHMOD b:*.*
```

for a system disk with a Volume ID of WP and a subdirectory name OVERLAY displays the following:

```
b:                    ibmbio.com          A S R H .
b:                    ibmdos.com          A S R H .
b:                  command.com           A . . . .
b:                          wp  (dir)     A S R H V
b:                     overlay  (dir)     . . . . .
5 files.
```

CLS

Purpose The CLS command clears the display screen.

Syntax CLS

Comments The CLS command is often used in batch processing files to clear the display before another command is begun.

Purpose

The COMP command compares the contents of a file or group of files with others to verify that they are identical.

Syntax

COMP [pathname1] [pathname2]

Comments

<pathname1> is the file or group of files that are to be compared with those specified in <pathname2>.

You must enter <pathname2> for this command to perform as expected.

More than one file may be specified if you use wild card characters. Only files that have matching names to those specified in <pathname1> are compared.

Files may be on either the same or different drives. Files may be in the same or different directory.

If the files to be compared are identical, the following message appears on your screen:

```
Files compare OK
Compare more files (Y/N)?
```

Press **N** to stop the COMP command.
Press **Y** to repeat the procedure with another set of files.

If more than a single file is being compared, the COMP process continues until all files have been compared.

The COMP command displays an error message if one of these problems occurs:

- a specified directory path is invalid
- two files to be compared are of a different size
- a file specified by pathname2 cannot be found.

An advisory message is displayed if in the same location of two compared files the data does not match. The message indicates the offset in bytes (within the files) and the contents of those sections of the files.

If ten data mismatches are found, the COMP command halts and displays this message:

```
10 mismatches - ending compare
```

If an End-Of-File marker is not found at the end of a COMP command process, this message appears on your screen:

```
Eof mark not found
```

You can ignore this message; it does not indicate an error.

CONFIG.SYS

Purpose

CONFIG.SYS is a special file that is processed automatically when DOS starts. For information about CONFIG.SYS, see **Appendix E.**

Purpose

The COPY command copies one or more files to another disk.

The COPY command also lets you add the contents of one or more files to the end of another file. This is called concatenating (joining) files.

Syntax

To COPY a file:

COPY [pathname] [pathname] [/V] [/A] [/B]

Comments

To COPY files:

If the source and destination files are in the working directory, you do not need to specify a complete pathname.

If the second pathname option is not given, the copy is sent to the default drive using the original file name (first pathname option). If the first pathname is on the default drive and you don't specify the second pathname, COPY halts. You may not copy a file to itself.

MS-DOS displays the error message:

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

If the second pathname entry is a drive designation, the file is copied using the original filename.

If the second pathname entry is a filename without a drive specifier, the original file is copied on the default drive with a different filename.

If the second pathname option is a complete file specification, the file is copied to the destination drive using the specified filename.

Options

/V MS-DOS verifies that the file is being correctly copied on destination disk.

When you use the /V option, the COPY command runs more slowly because of the verification process.

You would use this option whenever you want to be certain that important data

is being correctly recorded.

/A Indicates that the file is an ASCII file. This option entry applies to the filename it precedes and to all subsequent filenames on the command line until a /B is encountered.

In a source filename /A allows data to be copied up to, but not including an End-Of-File marker. (**CTRL Z** is the End-of-File marker for ASCII files.) No more data in the file is copied after a CTRL-Z is encountered. In a destination filename /A adds an End-Of-File character to the end of the copied file.

/B Indicates that the file is a binary data file. This option refers to the preceding filename and remains in effect until /A is encountered.

In a source filename /B copies the entire file, including an End-Of-File marker.

In a destination filename /B causes no End-Of-File marker to be added to the copied file.

To CONCATENATE files:

The COPY command lets you join files (concatenation) while copying. This is done by simply listing any number of files as options in the COPY command entry, each separated by a plus sign (+).

For example,

```
COPY MON.TX + TUE.TX APR22.TX
```

joins the files MON.TX and TUE.TX, copying them into the file APR22.TX on the default drive.

You can also use wild card characters to combine several files into a single file.

For example,

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

takes all of the files with the extension .LST and combines them into the file named COMBIN.PRN.

You can use wild card characters for more complicated file joinings. For example, the COPY command entry

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

joins each file with the extension .LST with a **corresponding** file with the extension .REF. A file named FILE1.LST is combined with a file named FILE1.REF (and so on) into a file named FILE1.PRN; a file named ABC.LST is combined with a file named ABC.REF to make a file named ABC.PRN.

This same technique can be used to combine all of the files with the extension .LST and the extension .REF into a single file by entering this command:

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

The resulting file is named COMBIN.PRN.

Be careful not to enter a concatenate COPY command where the destination file has the same extension as one of the source files. For example,

```
COPY  *.REF  ALL.REF
```

would result in an undetected error if ALL.REF already existed on the destination disk. At the point the error would be detected, the ALL.REF file would have been destroyed.

COPY compares the source filename with the destination filename. If they are the same, that single source file is skipped over. This error message is displayed:

```
Content of destination lost before copy
```

and the file joining process continues normally.

You can also "sum" files. For example,

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

appends all *.LST files, except ALL.LST itself, to ALL.LST. This command entry is the correct way to append files with the COPY command. No error message is displayed.

Purpose

The CTTY command lets you change the device used by MS-DOS to accept command input. ("TTY" is what MS-DOS calls your keyboard.)

Syntax

CTTY <device>

Comments

The <device> is the device from which commands are to be sent to MS-DOS. You use this command if you want to change the device where you are working.

For example, the entry

```
CTTY AUX
```

moves all command I/O (input/output) from the display keyboard to the AUX device such as a printer. The command

```
CTTY CON
```

returns command I/O back to the display keyboard. The valid legal device names are described in Chapter 2, **Reserved Filenames and Extensions.**

DATE

Purpose

The DATE command is used to enter or change the date used by MS-DOS to note in a directory when a file was recorded.

Syntax

DATE [mm-dd-yy]

Comments

If you type DATE, this message appears:

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

Press the **RETURN** key to leave the displayed date as shown.

If you type DATE followed by a date entry, the new date is used by MS-DOS. For example,

DATE 3-9-85

makes your computer use the date March 9, 1985.

A DATE entry must be entered using numerals only; letters are not permitted. The allowed options are:

mm	= 1-12
dd	= 1-31
yy	= 84-91 or 1984-1991

You can use hyphens (-) or slashes (/) to separate the date, month, and year entries. MS-DOS handles months and years correctly, whether the month has 31, 30, 29, or 28 days, including leap years.

Notes

You can change the date from within a batch processing file.

At the end of the year, the date WILL NOT AUTOMATICALLY CHANGE to the new year. Be sure to use the date command to make this change each year.

DEL (DELete)

Purpose

DELete is the same as ERASE. This command removes designated files from a directory.

Synonym

ERASE

Syntax

DEL [pathname]

Comments

Be careful. If you use wild card characters to describe a pathname, you may DELete more files than you intended. For example, the pathname

```
DEL A:*.*
```

causes the message

```
Are you sure?
```

to be displayed. If you type Y or y as a response, then **all** the files on that diskette are removed.

You can use ERASE instead of DELete.

Purpose

The DEBUG utility is an executable object program that resides on your MS-DOS System diskette. For information on DEBUG, see **Appendix D.**

DIR

Purpose

The DIR command displays all of the files in a specified directory.

Syntax

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

Comments

All the directory entries on the default drive are displayed if you only enter DIR.

If you enter a drive specification (DIR d:), all directory entries from the specified drive are displayed.

If you enter a filename without an extension (DIR filename), then all of the files in the directory with that filename are displayed.

If you enter a complete file specification (DIR d:filename.ext), each file in the directory that has that exact file specification is displayed.

Files are always displayed with their size in bytes, the time, and the date they were last modified.

You can use wild card characters with the

DIR command. See the chart below.

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

Options

There are two options with the DIR command.

/P selects Page Mode. With the /P option the directory pauses after the screen is filled. (A "page" is displayed.) To restart the display of the directory, press any key.

/W selects Wide Display. With the /W option only the filenames are displayed, five filenames per line. No other file information is displayed.

DISKCOMP

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.

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

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

Syntax

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

Comments

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.

DISKCOMP compares all tracks on a track-for-track basis and issues a messsage 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)?_
```

To end the command, press **N.**

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.

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).

Options

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

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

DISKCOPY

Purpose

The DISKCOPY command copies all of the files on the source disk to a destination disk. The diskettes must be in separate drives unless you have a single diskette drive computer.

Syntax

DISKCOPY [d:] [d:]

Comments

The first modifier is the source drive, the second is the the destination drive.

WITH A SINGLE DRIVE COMPUTER: You are prompted to insert diskettes at the appropriate times. DISKCOPY pauses between insertions until you press any key before continuing.

After copying, DISKCOPY prompts:

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

To copy another diskette, press **Y.**

To end the COPY, press **N.**

Notes

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

2. If you omit the second option, the default drive is used as the destination drive.

3. Disks that have had a lot of file activity (add and delete) are fragmented, because space on the disk is no longer sequential. The first free sector found is the next sector allocated regardless of its location.

   A fragmented disk performs slowly due to delays from finding, reading, or writing a file. Use the COPY command to copy a diskette that has had a lot of activity.

For example:

COPY A:*.* B:

copies all files from the disk in drive A: to the disk in drive B: sequentially, eliminating the fragmentation of space.

4. DISKCOPY automatically determines the number of sides to copy, based on the source drive and disk.

Purpose

The EXE2BIN command converts .EXE (executable) files to binary file format. Using EXE2BIN results in faster program loading and conserves space on the disk. For information about EXE2BIN, see **Appendix F.**

FDISK

Purpose

The FDISK command lets you set up a fixed disk drive to receive multiple operating systems.

Syntax

FDISK

Comments

FDISK lets you create and define the operating system partitions on your fixed disk.

Refer to Chapter 3, **Using a Fixed Disk,** for complete information.

FC
(File Compare)

Purpose

The File Compare utility compares the contents of two files.

Syntax

FC [/#][/B][/C][/W] <filename1> <filename2>

Comments

If you have copied a file and modified the copy, you may later want to compare copies to check the changes you have made. You can use File Compare to do this and direct that the differences between the two files be output to the screen or to a third file.

The files being compared may be either source files (files containing source statements of a programming language) or binary files (output from an assembler, a linker utility, or a high-level language compiler).

The comparisons are either on a line-by-line or a byte-by-byte basis. Line-by-line comparison isolates blocks of lines that are different between the two files and prints those blocks of lines. This is the default method. If there are too many differences (too many lines), the program simply reports that the files are different and stops. If no matches are found after the first difference, FC displays:

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

and returns to MS-DOS default drive prompt.

The byte-by-byte comparison displays the bytes that are different between the two files. For binary files larger than available memory, FC compares both files completely, overlaying the portion in memory with the next portion from disk. All differences are output in the same manner as those files that fit completely in memory.

Options

The following options are used with FC.

/# Number of lines required to match for the files to be considered as matching again after a difference has been found. # can be any number from 1 to 9. Default is 3. Use only in source comparisons.

/B Binary comparison of both files is performed. The two files are compared byte-by-byte with no attempt to re-synchronize after a mismatch. Mismatches are printed as follows:

```
--ADDRS----F1----F2-
xxxxxxx    yy    zz
```

where xxxxxxx is the relative address of the pair of bytes from the beginning of the file. Addresses start at 00000000; yy and zz are the mismatched bytes from file1 and file2 respectively.

If one of the files contains less data than the other, a message appears. If, for example, F1 ends before F2:

```
***Data left in F2***
```

/C Ignore case of letters. All letters in the files are considered upper case. Use only in source comparisons.

/W Compress whitespace (tabs and spaces) during comparison. Multiple contiguous whitespace in any line is considered as a single white space. Only beginning and ending whites are ignored. Use only in source comparisons.

Examples Assume these two ASCII files are on disk:

ALPHA.ASM	BETA.ASM
A	A
B	B
C	C
D	G
E	H
F	I
G	J
H	1
I	2
M	P
N	Q
O	R
P	S
Q	T
R	U
S	V
T	4
U	5
V	W
W	X
X	Y
Y	Z
Z	

One way to compare them is:

FC ALPHA.ASM BETA.ASM<cr>

FC compares the two files and displays the differences on the screen. All defaults remain intact. The output appears as follows.

```
————ALPHA.ASM
D                          NOTE: ALPHA file contains
E                          DEFG, BETA contains G
F
G

————BETA.ASM
G
```

```
————ALPHA.ASM
M                          NOTE: ALPHA contains MNO
N                          where BETA contains J12
O
P

————BETA.ASM
J
1
2
P
```

```
————ALPHA.ASM
W                          NOTE: ALPHA contains W
                           where BETA contains 45W

————BETA.ASM
4
5
W
```

If you use:

FC /B ALPHA.ASM BETA.ASM<cr>

The following binary comparison appears.

```
—ADDRS——F1——F2—
00000009    44    47
0000000C    45    48
0000000F    46    49
00000012    47    4A
00000015    48    31
00000018    49    32
0000001B    4D    50
0000001E    4E    51
00000021    4F    52
00000024    50    53
00000027    51    54
0000002A    52    55
0000002D    53    56
00000030    54    34
00000033    55    35
*** DATA left in F1 ***
```

Purpose

The FIND command searches for a specified string of text in a file or files.

Syntax

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

Comments

FIND is an MS-DOS filter. A filter discards all characters or values except those that match its selection criteria.

FIND accepts entries that are the object of the search (the text string) and the place or places (the file or files) where it is to look for matching data.

FIND displays all lines that contain the data being sought from the files specified in the command line.

Options

/V causes FIND to display all lines not containing the specified string.

/C causes FIND to print only the count of lines that contained a match in each of the files.

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

The text string must be enclosed in quotes. If a text string you are looking for is already enclosed in quotes, the entry on the command line must be in double quotes.

Examples

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

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

DIR B: ¦ FIND /V "DAT"

causes MS-DOS to display all names of the files on the disk in drive B: that do not contain the string DAT.

Purpose

FORMAT prepares diskettes or a fixed disk to receive data from your computer. Diskettes or a fixed disk must be formatted before they can be used.

Syntax

FORMAT [d:][/O][/V][/S][/1][/8]

Comments

FORMAT initializes the directory and file allocation tables on the specified drive. If no drive is specified, the diskette in the default drive is formatted.

Options

/O causes FORMAT to produce a PC DOS version 1.X compatible disk. The /O option causes FORMAT to reconfigure the directory with an OE5 hex byte at the start of each entry so that the disk may be used with 1.X versions of PC DOS, as well as MS-DOS 1.25/2.00 and PC DOS 2.00.

This option should only be given when needed because it takes a fair amount of time for FORMAT to perform the conversion, and it noticeably decreases 1.25 and 2.00 performance on disks with few directory entries.

/V causes FORMAT to prompt for a volume label after the disk is formatted.

/S If this option is used, it must be the last entry on the command line. /S copies the hidden files onto the diskette or fixed disk.

The files are copied in the following order:

IBMBIO.COM
IMBDOS.COM
COMMAND.COM

/1 causes the diskette to be formatted for single-sided use.

/8 causes the diskette to be formatted for use at 8 sectors per track. The default is 9 sectors per track.

Purpose

The GRAPHICS command lets you print any graphics displayed on the screen if your printer is compatible with the AT&T 473 graphics printer.

Syntax

GRAPHICS

Comments

After loading GRAPHICS into the memory of your computer, press and hold **SHIFT**, then press **PRT SC** to direct the graphics displayed on the screen to the graphics-compatible printer.

The GRAPHICS command defaults to a standard printer. Use the MODE command to set other parameters that are required for your own printer.

If color graphics are used, the screen image is printed in up to 4 shades of gray.

Use interrupt #5 (INT5) to print the screen from a program. Refer to the *System Programmer's Guide* for more information.

MKDIR
(MaKe DIRectory)

Purpose

The MKDIR command creates a new directory on the specified pathname.

Syntax

MKDIR <pathname>

Synonym

MD

Comments

You use MKDIR to create a hierarchical directory structure on your diskette or your fixed disk. You create subdirectories from your ROOT (top) directory by using the MKDIR command.

For example, the command entry

```
MKDIR \USER
```

creates a subdirectory\USER in your ROOT directory. To create a directory named JOE under **USER**, you enter:

```
MKDIR \USER\JOE
```

Refer to Chapter 3, **Using a Fixed Disk,** for more information about different ways to use directories and subdirectories to your best advantage.

Purpose

The MODE command lets you define some of the operating characteristics of your computer. There are three different uses of MODE.

Syntax

To set communication protocol for the RS-232C serial interface port:

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

To set the type of display you are using and how many characters are displayed on a single line:

```
MODE n[[,R|L],T]
```

To set printer parameters or to redirect printer output to the RS-232C serial interface port:

```
MODE LPT#: [chars][,spacing]
```

or

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

Comments

Setting Communication Protocols

MODE COMn:baud[,parity[,databits[,stopbits[,p]]]]

When you are setting the communication protocol for your serial interface port you must specify which port you are setting and the speed (baud rate) you want to use. Parity, databits, and stopbits entries are optional. You may also specify that the serial port continuously attempts to make and maintain contact with a printer or telephone coupler following time-out errors.

The syntax element **n** must be entered as either **1** or **2**. 1 is the built-in serial interface port. 2 is an optional second serial interface port.

The syntax element **baud** is the baud rate that the serial port uses to communicate. Only the first two digits of the baud rate need to be entered. The baud rates are: 110, 300, 600, 1200, 2400, 4800, or 9600.

Parity setting is optional. The entry may be either **e** (even parity), **o** (odd parity), or **n** (no parity). The default value if no entry is made is even parity.

Data bits are an optional entry. If no entry is made, the default value is 7. You may enter either 7 or 8 for the number of data bits in a character.

Stopbits are an optional entry. You may enter either 1 or 2 for the number of stopbits at the end of a character. If no stopbits entry is made, the default value is 2 stopbits if the baud rate is 110. The default value for all other baud rates is 1 stopbit.

The **p** entry in the command line is optional. If it is included, your serial interface port will continuously retry to send if time-out errors occur. Some types of printers may not match the timing requirements of your serial port. In most cases, the **p** option overcomes this problem.

Examples

MODE COM1:96

The form of the MODE command sets the baud rate to 9600 for the built-in serial interface port. All other entries are set to their default values.

MODE COM1:1200,N,1,P

This form of the MODE command sets the built-in serial port to 1200 baud, no parity, one stop bit, and continuous retry on time-outs. This command format might be used with a high-speed telephone coupler.

Comments

Setting Display Characteristics

MODE n[[,R|L],T]

Depending upon the type of display you are using with your computer, or if you have more than one display attached to your computer, you use this form of the MODE command to set the display type and format.

This form of the command requires an entry called an argument. Arguments are values that tell the computer what option you want.

Options

40 Sets the width of the display line to 40 characters.

80 Sets the width of the display line to 80 characters.

BW40 If you have a monochrome and a color display attached to your computer, this selects the color monitor to be active, sets the display mode to black and white and to 40 characters per line.

BW80 Same as above, except that 80 characters per line are displayed.

CO40 Same as BW40, except that the display mode is set to color.

CO80 Same as BW80, except that the display mode is set to color.

NOTE: The monochrome monitor on your AT&T PC 6300 is not a true monochrome monitor. It displays "shades of gray." If you have purchased applications software that supports color, mode CO80 will give you the most attractive display.

MONO If you have a monochrome and a color display attached to your computer, this selects the monochrome display. The width for monochrome displays is always 80 characters per line.

R Moves display one character to right.

L Moves display one character to left

/O Displays all the options that have already been selected by the MODE command.

Comments

Setting the Printer Characteristics and Selecting the Printer Port

MODE LPT#: (chars)[,spacing]

MODE LPT#:=COMn

LPT# is the device name for the parallel printer port(s) in your computer. The # is the printer port number you enter to select which printer port you are setting. The number you enter may be either 1, 2, or 3. Port #1 is the built-in parallel port in your computer and is the default port for all printing activity.

The first example above lets you set the number of characters per inch that your printer will print and the vertical spacing between lines of output. For example,

MODE LPT1: 12,6

sets parallel port #1, then passes the instructions to your printer to print 12 characters per inch and 6 vertical lines per inch.

Some printers do not accept this type of command. Check the user guide that came with your printer for more information.

In the second example above, you reassign the printer output port to be a serial interface port in your computer. You may want to do this

if you have a special printer attached to your computer that you use for correspondence-quality output or for graphics.

For example,

```
MODE LPT1:=COM2
```

redirects all printer output to an optional second serial interface port attached to your computer. COM1 is the name of the built-in serial port in your computer. COM2 would be an extra interface port that you may have added at a later time.

Note

In some cases you may need two MODE commands to set up the serial interface port for use with a printer. For example,

```
MODE LPT1: 2400,E,1,P
MODE LPT1:=COM1
```

would be required to set the printer port baud rate to 2400, even parity, 1 stop bit and continuous retry. The second MODE command redirects the output using the new settings to the built-in serial port in your computer.

MORE

Purpose

The MORE command limits the amount of information displayed on the screen to a single page at a time.

Syntax

MORE

Comments

The MORE command is an MS-DOS filter.

The MORE command takes input (such as a command from your keyboard) and displays one screen of information at a time. — MORE — is displayed at the bottom of your screen and the display pauses until you press **RETURN**.

This process continues until all of the input data has been read from the source file.

In the example below the piping symbol (|) is used to pipe the output from the text file to the MORE filter.

You use the MORE command to view a long file one screen at a time. For example

TYPE MYFILES | MORE

displays the file MYFILES (on the default drive) one screen at a time.

The MORE command can also be used with the output redirection character (>) to create a text file from the default device CON, your keyboard.

For example, the command entry

more> MYFILES

opens MYFILES on the default drive and lets you enter text from the keyboard. Pressing **CTRL C** ends the text entry and writes the text you've entered into the file.

LINK

Purpose

LINK is an executable program that combines object modules that are the output of the MACRO-86 assembler or a compatible compiler. For information about LINK, see **Appendix C.**

Purpose

The PATH command sets a search path for all MS-DOS file commands.

Syntax

PATH = [pathname[;pathname]...]

Comments

The PATH command lets you define where MS-DOS searches for external commands after it searches your current directory.

The default value is no path.

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

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

Every time you ask for an external MS-DOS command, MS-DOS searches \BIN\USER\JOE directory for external commands until you set another pathname or finish the current MS-DOS session.

MS-DOS will search more than one pathname if you separate the pathnames with semicolons. For example,

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

tells MS-DOS to search the \JOE subdirectory first, then the \SUE subdirectory, and last to search the \BIN\DEV\ subdirectory for the external commands.

MS-DOS searches the pathnames in the order specified in the PATH command.

The command PATH with no options displays the current pathname.

If you use a semicolon to specify PATH, MS-DOS searches only the current directory for the external commands.

Purpose

The PRINT command lets you print a text file on your printer while you are working with other MS-DOS commands. This is called "background printing" which means that the printing is being done in the background while the other work you are doing with MS-DOS is going on in the foreground.

Syntax

PRINT [[filename][/T][/C][/P]]

Comments

Use the PRINT command only if you have a line printer attached to your computer.

Options

The following options are used with the PRINT command.

/T TERMINATE stops the printing session and removes all files from the print queue that have not yet been printed.

/C CANCEL suspends printing of the specified file and all subsequent files until a /P option is entered.

/P PRINT begins printing. The preceding file and all subsequent files are added to the print queue until you enter a /C option.

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

Examples

PRINT /T

empties the print queue.

PRINT A:TEMP1.TST /C
A:TEMP2.TST A:TEMP3.TST

The /C cancels the printed output of the TEMP1.TST file and suspends the operation of the print queue until the /P is entered.

PRINT TEMP1.TST /C TEMP2.TST /P
TEMP3.TST

/C removes TEMP1.TST from the queue, /P adds TEMP2.TST and TEMP3.TST to the queue.

Purpose

The PROMPT command lets you assign the characters or phrase used as the MS-DOS prompt.

Syntax

PROMPT [prompt-text]

Comments

The PROMPT command lets you set the MS-DOS system prompt (A> or C>) to any other text you want.

If you don't enter any text with the PROMPT command, the prompt is reset to its default value, A> or C>.

You can set the prompt to a special prompt, such as the current time, by using the characters indicated below.

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

Specify This Character	To Get This Prompt:
$	The '$' character
t	The current time
d	The current date
p	The current directory of the default drive
v	The version number
n	The default drive
g	The '>' character
l	The '<' character
b	The '|' character
-	A CR LF sequence
s	A space (leading only)
h	A backspace
e	ASCII code X'1B' (escape)

Examples

PROMPT $n

Sets the default drive letter prompt.

PROMPT Time = t_Date = $d

Sets a two-line prompt that prints:

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

ANSI Escape sequences used by programmers can be used in the MS-DOS prompt. To use these sequences you must load ANSI. SYS into the computer. For example,

```
PROMPT $e[7m$n:$e[0m
```

Sets the prompts in inverse video mode yet keeps text in normal video mode.

For other uses of ANSI Escape sequences, see **Appendix G.**

RECOVER

Purpose

The RECOVER command lets you recover a file or an entire diskette or fixed disk drive that has bad sectors.

Syntax

RECOVER <filename | d:>

Comments

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

To recover a particular file, you enter

RECOVER <filename>

causing MS-DOS to read the file sector by sector and to skip the bad sector(s). If MS-DOS finds bad sector(s), the sector(s) are marked so that MS-DOS no longer puts any data in that sector.

To recover a diskette or fixed disk, enter

RECOVER <d:>

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

RENAME

Purpose

The RENAME command lets you change a file name to another file name.

Synonym

REN

Syntax

REN <pathname> <filename>

Comments

The first modifier <pathname> is the name of the file you want to change. The second modifier <filename> is the new filename you want to give to the file.

The first filename must include a drive designation if the file is on another drive than the default drive. The new filename is always created on the same drive or pathname as the source file.

You can use wild card characters in either option. All files matching the first filename are renamed. If wild card characters appear in the second filename, corresponding character positions are not changed.

For example, this command changes the names of all files with the .LST extension to similar names with the .PRN extension:

```
REN  *.LST  *.PRN
```

With this command, REN renames the file FLYER on drive B: to CRYER:

```
REN  B:FLYER  CR???
```

The file remains on drive B:.

RESTORE

Purpose

The RESTORE command restores to a fixed disk the files saved on diskettes by the BACKUP command.

Syntax

RESTORE <d:>
<d:>[pathname][filename][/S][/P]

Comments

The first modifier you specify is the drive designator of the disk containing the backed up files. The second parameter is the file specification indicating the files you want to restore.

The RESTORE command may not operate in the way you expect with files created with the BACKUP program supplied by other vendors.

Options

/S Restore subdirectories also.

/P Causes RESTORE to prompt you before restoring (writing over) files that have changed since they were last backed up, or files that are marked read-only. You then have the option of restoring the file or not.

NOTE: Read-only is a file attribute that an application can set by interfacing with MS-DOS internally or that you can set with the CHMOD command.

If you do not specify a pathname, only files in the current directory are restored. If you do not specify a filename, all files backed up from the directory are restored. You can put global characters in the filename and cause all of the files that match the filename to be restored.

Example

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

restores all files on the backup diskettes to fixed drive C:

```
RESTORE A: C:*.ASM
```

restores each file from the backup diskettes that has an extension of .ASM and that had originally been backed up from the current directory.

NOTE

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 prompts you to insert the next diskette.

If you backed up a file on more than one diskette, restore the file using the diskettes in the same order as they were saved.

If you used global filename characters, RESTORE continues to prompt you to insert the next diskette, even after it has restored all files on the backup diskette that match the specified filename, until it has processed all of the backup diskettes.

Exit codes

The RESTORE command will set the exit code number as follows:

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

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

RMDIR
(ReMove DIRectory)

Purpose

The RMDIR command removes a subdirectory from a hierarchical directory structure.

Synonym

RD

Syntax

RMDIR <pathname>

Comments

You use the RMDIR command to remove an **empty** directory. All the files must be first DELeted or ERASEd, except for the subdirectory shorthand symbols . and .. .

The single period (.) means the current directory. Double periods (..) mean the parent directory.

To remove the \BIN\USER\JOE directory, you first check the directory with a DIR and pathname to check that there aren't any important files that you want to keep, then you enter:

```
DEL  \BIN\USER\JOE\*.*
RMDIR  \BIN\USER\JOE
```

The directory has been deleted from the directory structure.

Purpose

The SET command is used to assign a value to an environment variable or to display the current values.

Syntax

SET [variable = string]

Comments

The SET command is used if you want to set values that will be used by programs or .BAT files.

For example,

```
SET TTY=VT100
```

sets the TTY variable to have the value VT100 until you change it with another SET command.

If you type SET with no variables, MS-DOS displays the current values of the environment variables.

More information about using the SET command appears in Chapter 6, **Batch Processing Commands**.

SHIP

Purpose

The SHIP command positions the heads on the internal hard disk for shipment.

Syntax

SHIP

Comments

After you enter SHIP command, turn off the system immediately, since using the hard disk moves the heads out of the shipping position.

Use this command when you move a system containing an internal hard disk.

Refer to the manufacturer's instructions on moving any external hard disk.

Purpose Lists filenames and file sizes of specified files. Shows the total number of files and total space allocated for the files.

Syntax SIZE [d:][pathname][filename] [d:]

Remarks Use SIZE with a pathname or filename. Global characters are permitted. If no parameter is entered, SIZE defaults to *.* of the current directory.

Examples

To list all files with their sizes, type:

```
SIZE *.*
```

You can use SIZE to display the names and sizes of files in a particular directory. For example, to produce a list of all the files with the extension of COM in subdirectory \BIN, you would type:

```
SIZE \BIN\*.COM
```

This produces the following:

```
chkdsk.com      6400
diskcopy.com    2576
format.com      6912
comp.com        2534
4 files, total of 18422 bytes.
```

If a drive letter is specified as the second argument, the cluster size on the specified drive is used to determine space requirements.

```
SIZE *.* B:
```

This form of the SIZE command lists exact file sizes, the amount of disk space on drive B needed by each file, and the total amount of disk space needed for all files.

If a drive letter is specified as the second argument, the cluster size on the specified drive is used to determine space requirements.

Purpose

The SORT command is used to sort the contents of a file and send the result to a device or file you select. The file sorted can also be a directory or information that would be displayed on your screen.

Syntax

SORT [/R] [/+n]

Comments

Use the symbols < and > to direct how SORT is to accept and send information. The symbol "<" means "accept data from this file" and the symbol ">" means "send the output data from the command to this file."

You can use SORT to alphabetize a file by a specified column, such as one of the file statistic columns in a DIRectory listing.

Options

The following options are used with the SORT command.

/R SORT is done in ASCII order unless you specify the /R option. This option reverses the sort; that is, sorts from Z to A.

/+n SORT starts with column n where n is some number. If you do not specify this option, SORT begins sorting with the first column of input.

Examples

The following command reads the file UNSORT.TXT, reverses the sort order, then writes the result to a file named SORT.TXT:

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

The following example sends (using the "pipe" symbol |) the DIR command listing to the SORT filter. The SORT command sorts the directory listing starting with column 14 (the column that contains file size in the DIR listing), then sends the result to the screen.

The result of this command is a directory sorted by file size:

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

The command

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

does the same thing as the previous example, except that the MORE filter lets you read the sorted directory listing one screen at a time.

Purpose

The SYS command copies the MS-DOS System files to the specified disk drive from the default drive.

Syntax

SYS <d:>

Comments

You use SYS to update MS-DOS or to place system files on a formatted disk that does not already have any files on it.

Your destination disk **must be completely blank** or already have the MS-DOS files IBMBIO.COM and IBMDOS.COM. The files are copied in the following order:

IBMBIO.COM
IBMDOS.COM

They are both hidden files that don't appear when the DIR command is performed. COMMAND.COM (the command processor) is **not** transferred with a SYS command. Use the COPY command to move the COMMAND.COM file to the destination diskette.

Be careful. Diskettes that contain an earlier version of the MS-DOS System files are probably not the same size as the System files you transfer using the SYS command.

It is safer to use the /S option with the FORMAT command to place a current version of the System files on a new diskette, then COPY all data and program files over to the new diskette.

Purpose

The TIME command lets you display and set the time of day used by your computer to mark when files were last written or changed.

Syntax

TIME [hh[:mm]]

Comments

If only the TIME command is entered without any values for the current time, your computer displays:

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

If you don't want to change the time shown, press the **RETURN** key.

You may enter a new value for the time of day at any time by simply typing the TIME command and a value. For example,

```
TIME 8:20
```

sets the new time of day to 8:20am.

The value for TIME is entered using numerals only; letters are not allowed. The valid values are:

```
hh      = 00-24
mm      = 00-59
```

The hour and minute entries must be separated by colons. You cannot set the seconds or hundredths of seconds values.

MS-DOS uses the time entered as the new time if the values and separators are valid. If the values or separators are not valid, MS-DOS displays the message:

```
Invalid time
Enter new time:_
```

Make a valid entry for the current time and press the **RETURN** key.

Purpose

Displays directories and files and optionally deletes files throughout the entire tree directory structure.

Syntax

TREE [d:][filename] [/A][/D][/Q]

Remarks

Use TREE to display files and subdirectory structure. The display starts with the current directory and displays all subdirectories down through the tree directory structure.

You can specify a disk drive in the usual manner. TREE D: lists all the directories on drive D.

TREE has three options that change the way it works. These are:

/A	Archive files
/D	Delete files
/Q	Question before every delete

If you specify /A after a filename, TREE only lists files that have the A, or Archive, attribute set. Whenever you modify file in any way, MS-DOS sets the A attribute to mark it as "changed." Programs such as BACKUP use this in determining which files need to be backed up. You can determine which files you have changed by typing (from the ROOT):

```
TREE *.* /A
```

/D and /Q work together. A /D after the filename tells TREE to delete each file it finds. A typical use for this is for deleting all .BAK files from hard disk with many subdirectories, where it would be tedious to enter the name of each directory manually.

/Q causes TREE to ask "Delete? (y, n)" before actually deleting the files it finds.

TREE *.BAK /D

Deletes all .BAK files found

TREE *.BAK /D/Q

Deletes each .BAK file found only if you type 'y' at the "Delete" question.

The examples will use the following directory structure of a disk called SYSTEM.

```
\ROOT
  COMMAND.COM
  AUTOEXEC.COM
  \DOS
    DEBUG.COM
    SYS.COM
  \BIN
    EDITOR.COM
    LETTER.TXT
    MEMO.TXT
  \BASIC
    BASICA.COM
    SINE.BAS
```

Examples

TREE

entered with no parameters from the ROOT directory would display:

```
DOS
BIN
BASIC
```

To display the directory structure of all files within the current directory, type:

TREE *.*

which produces:

```
command.com
autoexec.bat
DOS
  debug.com
  sys.com
BIN
  editor.com
  letter.txt
  memo.txt
BASIC
  basica.com
  sine.bas
3 subdirectories, 9 matching files.
```

To display all directories that contain files with a particular extension, type:

```
TREE *.TXT
```

which displays:

```
DOS
BIN
  letter.txt
  memo.txt
BASIC

3 subdirectories, 2 matching files.
```

The following is an example of using TREE to delete files. If you have a file DATE.BAS in several subdirectories on your disk, entering the following deletes them if you answer "y" to the "Delete" question. For example, type:

```
TREE date.bas /d/q
```

which produces:

```
DOS
BIN
  date.bas delete? (y,n) deleted
BASIC
  date.bas delete? (y,n) deleted
3 subdirectories, 2 matching files.
```

TYPE

Purpose

The TYPE command displays the contents of a file on the screen.

Syntax

TYPE <filename>

Comments

You use the TYPE command to check the contents of a file without modifying it or needing to use another program to display the contents of a file.

The file that you process with a TYPE command should be an ASCII text file, otherwise the results of this command are unpredictable and usually result in a garbled display that does not give you enough information to determine what the file contains. Binary files that contain control characters will produce this type of garbled display.

Use the MORE command with a TYPE command to view a screen page at a time.

The only screen formatting command that the TYPE command recognizes and acts upon is TAB. TABs are set at every eight columns.

Purpose

The VER command displays the version of MS-DOS that is in memory in your computer.

Syntax

VER

Comments

If you are working with different versions of MS-DOS you use the VER command to determine which you are currently using.

Some MS-DOS commands (DISKCOMP, for example) require that the diskettes be the same MS-DOS version in order to perform the task you request.

VERIFY

Purpose

The VERIFY command is an option that determines whether any write to disk you have performed was done correctly.

Syntax

VERIFY [ON|OFF]

Comments

The VERIFY command is the same as the /V option in the COPY command. If you want to verify that all of the files you copy are correctly written to the destination diskette or fixed disk drive, set the VERIFY command to an ON status.

When you set VERIFY to ON MS-DOS performs a VERIFY each time you use any MS-DOS command. An error message is displayed if MS-DOS is unable to successfully write the file to the destination diskette or fixed disk.

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

Purpose

The VOL command displays the label, if any, of the specified diskette.

Syntax

VOL [d:]

Comments

VOL [d:] displays the volume label of the disk in drive d:. If you don't specify [d:] MS-DOS displays the label of the diskette in the default drive.

A diskette or fixed disk volume label is initially set with the /V option of the FORMAT command.

6 Batch Processing Commands

- Batch Processing Commands
- Stopping Batch Processing
- Parameters in Batch Files

Batch Processing Commands

In your daily work, you may find yourself typing the same sequence of commands time and again to perform some frequently used task. Batch processing lets you combine a series of commands into a special file called a batch file, and execute the entire sequence by typing just the name of the batch file. The commands in a batch file are executed as if they had been typed at the keyboard. A batch file must be named with the .BAT extension and it is executed by typing the batch file-name without the extension.

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

A .BAT file may contain any MS-DOS command. The special MS-DOS commands described in this chapter let you control the tasks your batch processing files perform more closely.

Here are some additional uses for batch command files:

- Automatically setting the MODE for your printer, monitor and communications lines. (Use AUTOEXEC.BAT)
- Automatically going to a particular directory when you start up the system. (Use AUTOEXEC.BAT)
- Automating a backup procedure to assure that the correct set of files is always backed up.

How to Create an AUTOEXEC.BAT File

If, for example, you wanted to automatically load GWBASIC and run your inventory program (called INVNTORY) every time you started MS-DOS, here is how you would create the AUTOEXEC.BAT file for that purpose:

1 If your current directory is not the ROOT directory, go back to the ROOT directory using the CHDIR command. AUTOEXEC.BAT must reside in the ROOT directory so that MS-DOS can find it when you start your system.

2 Type:

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

This statement tells MS-DOS to copy the information you enter through the keyboard (CONsole) to the AUTOEXEC.BAT file.

3 Now type:

```
GWBASIC INVNTORY
```

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

4 Press the **CTRL** and **Z** simultaneously, then press the **RETURN** key.

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

Here is an example of another AUTOEXEC.BAT file:

```
MODE COM1:12,e,7,1,p
MODE LPT1:=COM1
DIR /W
PROMPT $p$g
```

The first line of the file sets up communications port 1 to operate at 1200 baud, at even parity, with 7 data bits and 1 stop bit, and to retry continuously in case of a time-out error. This port will have a serial printer.

The second line redirects the output that would normally go to printer LPT1 to COM1, the serial printer.

The third command directs MS-DOS to display a directory of the default disk, in the wide format.

The fourth line tells MS-DOS to display the name of the current directory instead of the normal prompt. For example, if the current directory was "work" on drive B, MS-DOS would prompt you with

```
work>
```

instead of

```
B>
```

Stopping Batch Processing

Batch processing files are stopped by pressing **CTRL C** or **CTRL SCR LOCK**. When you stop a batch file, the message

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

appears on your screen. If you want to stop the batch file, press **Y**. If not, press **N**, and the task the batch file is performing continues.

Parameters in Batch Files

Often you have repetitive tasks which are quite similar—perhaps differing only by the file being processed. A parameter is used to provide a file name to the .BAT file. This way, a batch file can immediately perform a task for you without having to be specially edited to match your current needs.

Thus UPDATE, a .BAT file, could be run twice weekly. On Wednesday night:

```
A>UPDATE.BAT  MON.SLS  TUES.SLS  WEDS.SLS
```

And on Saturday night:

```
A>UPDATE.BAT  THURS.SLS  FRI.SLS  SAT.SLS
```

Up to 10 parameters may be used. They are named %0,%1 . . . . %9. Parameter values are located on the .BAT command line. Parameters must be separated by spaces.

```
A>%0[.BAT] %1 %2 . . . . %9
```

%0 is always the name of the .BAT file.

Variables in Batch Files

Variables provide another way of passing specific data to a batch processing file.

There is an important difference between variables and parameters. A variable, once declared, retains its value until you either reset it or turn the computer off. Thus .BAT files pass values to other .BAT files and to the system.

A variable is a text string (a name) enclosed by % signs. For example, %variable%.

Values are given to variables with the MS-DOS SET command. For example, if your batch file contains the statement

```
LINK %FILE%
```

you can set the name that MS-DOS uses for that variable with the SET command. By entering

```
SET FILE=DOMORE
```

the %FILE% variable is replaced with the filename DOMORE.

Purpose

The batch command ECHO lets you choose whether or not to have the commands in your batch file displayed on your screen as each command is performed.

Syntax

ECHO [ON|OFF|<message>]

Comments

Commands in a batch file are normally displayed as they are performed by your computer. If you enter ECHO OFF into a batch file the commands are not displayed as they are performed. ECHO ON causes the commands to be displayed when they are performed.

If neither ON or OFF is specified, entering ECHO displays the current setting on your screen.

If ECHO is set to OFF, you can use ECHO <message> to display text on your screen. You use this only in a batch processing file.

FOR

Purpose

The FOR command is used to expand the power of a batch or interactive command.

Syntax

Batch processing:

FOR %%<c> IN <set> DO <command>

Interactive processing:

FOR %<c> IN <set> DO <command>

Comments

%%<c> is a variable. <c> can be any character except 0,1,2,3...,9. This avoids confusion with the %0-%9 batch parameters.

<set> is a list of items separated by spaces and enclosed in parentheses. For example

(item1 item2 item3 ... itemN)

is a <set>.

The %%<c> variable is set sequentially to each member of <set>, and then <command> is performed. If a member of <set> is an

expression involving either * or ?, or both, then the variable is set to each matching pattern from the file.

In this case, only one such <item> may be in the set. Any <item> other than the first <item> is ignored.

Examples

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

```
FOR %%f IN (BAK ART BUDGT) DO REM %%f
```

The '%%' is needed so that after batch parameter (%0-%9) processing is complete, a '%' remains. If only '%f' was entered, the MS-DOS batch parameter processor sees the first '%', looks at 'f', decides that '%f' is a bad parameter reference, and discards the '%f'. The FOR command would never receive this parameter. In a batch file, you must use the expression '%%'.

If you are in interactive MS-DOS processing mode, only **one** '%' is needed.

You cannot nest FOR commands in MS-DOS like a FOR-NEXT command is used in GW BASIC. If you try to do this, the message

```
FOR cannot be nested
```

appears when you are running your batch program. The program will not perform as you expected.

Purpose

The GOTO command is a batch file-only command used to direct your program to perform the command following the item described in the command modifier.

Syntax

GOTO <label>

Comments

A <label> must be included in the GOTO command entry. <label> is the place in your batch file that you want the program to jump to, then perform the commands that **follow** that command.

For example,

```
:spot
REM This is a loop
GOTO spot
```

causes an infinite sequence of messages to be displayed:

```
A>REM This is a loop

A>GOTO spot

A>REM This is a loop

A>GOTO spot

A>REM This is a loop
.
.
.
etcetera
```

If you do not include <label> in the GOTO command, the batch processing file stops.

Any line in a batch processing file that begins with ':' is treated as a label but otherwise ignored.

Purpose

The IF command allows conditional processing of MS-DOS commands.

Syntax

IF <condition> <command>

Comments

When <condition> is true, then the MS-DOS <command> is performed.

When <condition> is not true, the <command> is ignored and the next line in the batch processing file is performed.

<condition> must be one of the following:

ERRORLEVEL <number>

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

<string1> = = <string2>

True if and only if <string1> and <string2> are identical after parameter substitution. Strings may not contain any punctuation marks that MS-DOS uses for other purposes.

EXIST <filename>

True if and only if <filename> exists.

NOT <condition>

True if and only if <condition> is false.

Examples

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

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

Purpose

The PAUSE command lets you stop the batch processing until some action has happened.

Syntax

PAUSE [comment]

Comments

Use the PAUSE command to suspend processing of a batch file. For example, you may want to change diskettes. A PAUSE command stops your program to allow you to do so.

When the command processor encounters PAUSE, it displays:

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

If you press **CTRL C** or **CTRL SCR LOCK**, this prompt appears:

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

Type Y to stop the batch file and return to MS-DOS command level. You can use the PAUSE command to segment a batch file into segments that can be stopped at any appropriate point.

<comment> is optional and should be entered on the same line as PAUSE. <comment> is used to prompt — with a meaningful message — the batch file user to take some action when the file pauses.

The <comment> is displayed **before** the "Strike a key when ready . . ." message.

Purpose

The REM command lets you see any text that is on the same line as the REM command when a batch processing file is being run.

Syntax

REM [comment]

Comments

Use only the space, tab, and comma characters in the text of a REM comment. Other punctuation may be interpreted as a drive specifier or part of an MS-DOS command.

Example:

```
REM This file checks new disks
REM It is named NEWDISK.BAT
PAUSE Insert new disk in drive B:
FORMAT B:/S
DIR B:
CHKDSK B:
```

SHIFT

Purpose

The SHIFT command lets your batch file access more than 10 replaceable parameters.

Syntax

SHIFT

Comments

MS-DOS allows 10 parameters (%0-%9) to be used by a batch file. SHIFT lets you move more parameters into a queue, discarding the first parameter.

Additional parameters (more than ten) should be on the same command line in order to be shifted into the queue.

The SHIFT command works like this:

```
if %0 = "bak"
%1 = "art"
%2 = "budgt"
%3...%9 are empty
```

then a SHIFT results in the following:

```
%0 = "bak"
%1 = "budgt"
%2...%9 are empty
```

The %0 parameter is always the name of the current batch processing file and is not affected by a SHIFT command.

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

APPENDICES

A Messages

This appendix contains an alphabetical listing of the messages that MS-DOS displays when it is unable to perform a requested task.

Most of the messages you may encounter when using MS-DOS refer to problems in reading from or writing to devices such as disk drives or printers. These messages are called device errors.

Another form of message appears as a result of problems with a diskette or disk drive. These are called disk errors.

Other messages that you may encounter from time to time are related to the specific command or task that you are trying to perform. Included in the following list of messages, is the name of the command associated with the displayed message.

Device Errors

Device error messages all take the same form. They are displayed as:

```
<type> error reading/writing <device>
Abort, Retry, Ignore?_
```

where <type> represents one of the following:

Bad call format
Bad command
Bad unit
Data
Disk
No paper
Non-DOS disk
Not ready
Read fault
Sector not found
Seek
Write fault
Write protect

When you receive one of these messages, you have the choice of doing one of the following:

- enter **A** for abort. This ends the program that requested the read or write.
- enter **R** for Retry. This causes your computer to reattempt the operation. Correct the problem that caused this message, if you can, and press **R** for Retry.
- enter **I** for Ignore. This causes your computer to ignore the problem and attempt to continue the program. If you choose this option, you may lose some data. Be careful.

These device error messages are described along with the rest of the error messages in the following list of MS-DOS messages.

Disk Errors

If a disk read or write error occurs at any time during a command process or when you are running an application program, MS-DOS displays an error message in the following format:

```
<yyy> ERROR WHILE <I/O action> ON DRIVE <x>
Abort, Ignore, Retry:_
```

In this format, <yyy> may be one of the following:

WRITE PROTECT
BAD UNIT
NOT READY
BAD COMMAND
DATA
BAD CALL FORMAT
SEEK
NON-DOS DISK
SECTOR NOT FOUND
NO PAPER
WRITE FAULT
READ FAULT
DISK

<I/O action> is either READING or WRITING

The drive designation <x> is the drive where the problem occurred.

Enter either **A, I,** or **R** as noted above in order to proceed with your choice of action.

Allocation error, size adjusted (CHKDSK)

The file allocation table contains an invalid sector number. The file is truncated at the end of the last valid sector. CHKDSK has automatically performed the only remedy available at this point.

Bad call format error (device error)

A request header of incorrect length was passed to a device driver. Contact your dealer for more information.

Bad command error (device error)

A device driver has issued an invalid command to the named device. Contact your dealer for more information.

Bad unit error (device error)

An invalid subunit number has been sent to a device driver. Contact your dealer for more information.

Cannot CHDIR to root
Processing cannot continue (CHKDSK)

The disk you are checking is faulty. Restart MS-DOS and try to RECOVER the disk.

Cannot do binary reads from a device (COPY)

You have tried to use the /B option with the name of a device. Place an /A option after the device name to copy in ASCII mode.

Cannot edit .BAK file - rename file (EDLIN)

You have tried to edit a file that has a .BAK extension. Either edit instead the more up to date version of the file, or rename the .BAK file before attempting to edit it.

Data error (device error)

Data could not be read/written correctly because of a faulty disk. Restart MS-DOS and try to RECOVER the disk.

Disk error (device error)

An error has occurred reading from or writing to a disk. Attempt to correct the problem and reattempt this operation.

Disk error writing FAT (CHKDSK)

An error occurred while CHKDSK was trying to update the file allocation table. Copy the diskette and reattempt the procedure. If this does not work, try to RECOVER the disk. If this does not work, the diskette is irreparably damaged.

Disk full - file write not completed (EDLIN)

The diskette you are using does not have enough space to save all of the file. Part of the file may have been saved on the diskette, but the remainder that has not been saved is lost. This is an unrecoverable error. Repeat your edit using a diskette with sufficient storage space.

Drive not ready (PRINT)

If this message appears while PRINT attempts a disk access, PRINT keeps trying until the drive is ready. Any other error cancels the current file. In this case, an error message is printed on your printer.

Duplicate file name or file not found (RENAME)

An attempt has been made to rename a file with a file-name that already exists on the destination disk, or the file to be renamed could not be found on the specified disk drive.

Entry error (EDLIN)

You have incorrectly entered an EDLIN command. Reenter the command.

File cannot be copied into itself (COPY)

A request was made to COPY a file and place the copy (with the same name) in the same directory as the source file. Either change the name given to the copy or put it on another diskette or directory.

File creation error (MS-DOS and commands)

An unsuccessful attempt was made to add a new file to the directory. Run CHKDSK to determine the cause of the error.

Filename is cross linked on cluster (CHKDSK)

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

Filename must be specified (EDLIN)

You did not specify a filename when you started EDLIN. Specify a filename.

File not found (EDLIN)

The file you named in the Transfer command does not exist. Check your file name selection and try again.

File not found (MS-DOS and commands)

The file named in a command does not exist on the disk in the specified drive. Check your filename entry or drive selection and try again.

FIND: File not found <filename> (FIND)

The file you specified does not exist on the drive. Check your filename entry or drive selection and try again.

FIND: Invalid number of parameters (FIND)

A string was omitted when specifying a FIND command. Reenter the command with the search string.

FIND: Invalid parameter <option-name> (FIND)

An invalid parameter was entered with the FIND command. Reenter the command with a valid parameter.

FIND: Read error in <filename> (FIND)

An error occurred when FIND tried to read the file specified in the command. Reattempt the command after correcting an obvious problem (disk drive door open, no disk, etc.). Try to RECOVER the file.

FIND: Syntax error (FIND)

You entered an illegal string with the FIND command. Reenter the correct string.

First cluster number is invalid, entry truncated (CHKDSK)

An invalid pointer to the data area has been found in the file whose name precedes this message. If /F was specified, the file is truncated to zero length. This is an irrecoverable error.

Fixups needed-base segment (hex): (EXE2BIN)

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

Incompatible system size (SYS)

The hidden files IBMBIO.COM and IBMDOS.COM do not take up the same amount of space on the target diskette as the new system needs. Use the same version of MS-DOS to move these files.

Insufficient disk space (MS-DOS and commands)

There is not enough free space on the diskette to save the file or perform the MS-DOS operation. Use another diskette. You may have lost some data as a result of the operation that caused this message to appear.

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

CHKDSK cannot create an entry in the root directory for saving lost chains as files (see message "X lost clusters found in Y chains/ Convert lost chains to files(Y/N)?") because the root directory is full. You should copy some files from the root directory to another disk, then enter another CHKDSK command.

Invalid current directory
Processing cannot continue (CHKDSK)

CHKDSK has found an error in the disk's current directory. Restart your computer and rerun CHKDSK.

Invalid date (DATE)

You have tried to enter an invalid date. Review the DATE command parameters in Chapter 5 and reenter the date.

Invalid drive name or file (EDLIN)

You did not specify a valid drive or filename when you started EDLIN. Check your entry and try again.

Invalid drive specification (MS-DOS and commands)

You have entered an invalid drive specifier in a command. Check your entry and try again.

Invalid number of parameters (commands)

You have entered the wrong number of parameters on the command line. Refer to the syntax of the command you have selected and try again.

Invalid parameter (commands)

You have entered an incorrect parameter. Refer to the syntax of the command you have selected and try again.

Invalid subdirectory (CHKDSK)

The subdirectory contains invalid information. Rerun the CHKDSK command with the /V option for further information.

Invalid time (TIME)

You have entered an invalid time or the wrong punctuation mark. Refer to the TIME command syntax and try again.

Line too long (EDLIN)

During the (R)eplace command operation, the string given as the replacement caused the line to expand beyond the 253-character limit. The (R)eplace command has not been properly performed. Break the long line into shorter lines and try again.

List output is not assigned to a device (PRINT)

You have specified an invalid device. Specify a valid list device.

Must specify destination number (EDLIN)

You have not specified a destination number for a (C)opy or (M)ove command. Enter a destination number when you try again.

No files match <filespec> (PRINT)

You have entered a file specification for files to add to the print queue, but no files match your entry. Check your filenames and try again. NOTE: If there are no files in the print queue, a message does not appear.

Non DOS disk error (device error)

The file allocation table on the diskette contains invalid information. If you have inserted the correct diskette, it must be reFORMATted. Attempt to copy important files from this diskette to another before you FORMAT this diskette.

No paper error (device error)

Your printer is not powered ON or it is out of paper. Check your printer and try again.

No room for system on destination disk (SYS)

There is not enough room on the target diskette for the IBMBIO.COM and IBMDOS.COM files.

No room in directory for file (EDLIN)

The directory of the specified disk is already full, or the specified disk drive or filename is illegal. Use another diskette or check your entry and try again.

Not enough room to merge the entire file (EDLIN)

There is not enough room in your computer's memory to enable a Transfer command to merge the entire contents of the files. Break the files into smaller files and try again.

Not ready error (device error)

The device you've selected is not ready for the read/write operation you've requested. Check the device and try again.

Out of environment space

You have attempted to access the PATH command from GWBASIC. This is not allowed.

PRINT queue is empty (PRINT)

There are no files in the print queue. Files must be placed in the print queue before they can be printed.

PRINT queue is full (PRINT)

You cannot place more than ten files in a print queue.

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

The disk you are using is a non-MS-DOS diskette. Indicate whether or not you want CHKDSK to continue.

Read fault error (device error)

MS-DOS cannot read the requested data from the named device. Check your entry and try again.

Read error in <filename> (commands)

The command you've entered could not read the entire file. Try again. If the problem persists, try to RECOVER the file.

Sector not found error (device error)

The sector containing the requested data cannot be found or cannot be read. This is usually due to a bad sector on the diskette or disk. Try to RECOVER the diskette. If this does not solve the problem, the data is lost.

Seek error (device error)

The diskette drive cannot find the proper track on the diskette. Try again. If the problem persists, the diskette is probably damaged and cannot be recovered.

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

To convert the problem directory to a file, press **Y**. You can then repair or delete the directory at a later time.

Write fault error (device error)

MS-DOS cannot successfully write data from or to the named device. Check your entry and try again.

Write protect error (device error)

You have tried to write data onto a diskette that is write-protected. Check the diskette to be sure that you want to write to or change information on this diskette.

X lost cluster found in y chains
Convert lost chains to files (Y/N)?

CHKDSK has found lost clusters on your diskette. These may be parts of data files that have somehow been separated from their original files. If you press **Y**, these clusters are written into files that are placed in the root directory under the name(s) FILEnnnnnn. You can then attempt to repair damaged files or delete these safety files.

CHKDSK then displays:

```
X bytes disk space freed
```

If you select **N** and have not specifed the /F option of CHKDSK, the clusters are freed (deleted), and the message

```
X bytes disk space would be freed
```

appears on your screen.

B EDLIN The Line Editor

- Introduction
- Command Information
- EDLIN Commands
- How To Start EDLIN
- Special Editing Keys
- Command Options
- Error Messages

Introduction

This appendix is a function and command reference to the EDLIN program on your System diskette. EDLIN is provided for use by persons with prior experience in programming computers. EDLIN is not intended for novice users. If you are new to using computers, you will find that a dedicated word processing program may be easier to use. The information in this appendix does not include a tutorial for use by beginners.

If you are an experienced computer user you may choose to use EDLIN to create, change, and display files, whether they are source program or text files.

You can use EDLIN 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 in files created or edited by EDLIN is divided into lines, each up to 253 characters long. 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 being inserted.

When you delete lines in a file, all line numbers following the deleted text decrease automatically by the number of lines deleted. As a result, lines are always numbered consecutively in your file.

Command Information

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

1 Pathnames are acceptable as options to commands.

For example, typing EDLIN \BIN\USER\JOE \TEXT.TXT allows you to edit the TEXT.TXT file in the subdirectory \JOE.

2 You can reference line numbers relative to the current line (the line with the asterisk). Use a minus sign with a number to indicate lines before the current line. Use a plus sign with a number to indicate lines after the current line.

Example:

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

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

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

Examples:

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

15;-5,+5L

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

SThis string **CTRL Z**-5,+5L

4 You can type EDLIN commands with or without a space between the line number and command.

For example, to delete line 6, the command 6D is the same as 6 D.

5 It is possible to insert a control character (such as CONTROL-C) into text by using the quote character CONTROL-V before the capital letter associated with it while in insert mode. CONTROL-V tells MS-DOS to recognize the next capital letter typed as a control

character. It is also possible to use a control character in any of the string arguments of Search or Replace by using the special quote character. For example:

S **CTRL V** **Z**

finds the first occurrence of CONTROL-Z in a file

R **CTRL V** Z **CTRL Z** foo

replaces all occurrences of CONTROL-Z in a file by foo

S **CTRL V C CTRL** **Z** bar

replaces all occurrences of CONTROL-C by bar

It is possible to insert CONTROL-V into the text by typing CONTROL-V-V.

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

EDLIN Commands

The EDLIN commands are summarized in the following table. They are also described in further detail following the description of command options.

Command	Purpose
<line>	Edits line no.
A	Appends lines
C	Copies lines
D	Deletes lines
E	Ends editing
I	Inserts lines
L	Lists text
M	Moves lines
P	Pages text
Q	Quits editing
R	Replaces lines
S	Searches text
T	Transfers text
W	Writes lines

How To Start EDLIN

To start EDLIN, type:

EDLIN <filename>

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

```
New file
*_
```

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

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

If you want to edit an existing file, <filename> should be the name of the file you want to edit. When EDLIN finds the file you specify on the designated or default drive, the file is loaded into memory.

If the entire file can be loaded, EDLIN displays the following message on your screen:

```
End of input file
*
```

You can then edit the file using EDLIN editing commands. If the file is too large to be loaded into memory, EDLIN loads lines until memory is 3/4 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 save some of the edited lines on disk to free memory; then EDLIN can load the unedited lines from disk into memory. Refer to the Write and Append commands in this Appendix for more information.

When you complete the editing session, you can save the original and the updated (new) files by using the End command. The End command is discussed in this appendix in the section **EDLIN Commands**. The original file is renamed with an extension of .BAK, and the new file has the filename and extension you specify in the EDLIN command.

The original .BAK file is not erased until the end of the editing session, or until disk space is needed by the editor (EDLIN).

Do not try to edit a file with a filename extension of .BAK because EDLIN assumes that any .BAK file is a backup file.

If you find it necessary to edit such a file, rename the file with another extension (using the MS-DOS RENAME command discussed in Chapter 5), then start EDLIN and specify the new <filename>.

Special Editing Keys

The table below describes some of the commands, codes, and functions that are assigned to the special editing keys.

The "command line" is the line on your screen where you enter the text and EDLIN commands for the line you are editing. A "template" is the line you create, in part, from the line you are editing.

Special Editing Keys

Function	Key	Description
Copy 1 character	**F1**	Copies 1 character from the template to the new line.
Copy up to character	**F2**	Copies all characters from the template to the new line, up to the character specified.
Copy template	**F3**	Copies all remaining characters in the template to the screen.
Skip one character	**DEL**	Does not copy (skips over) a character.

Skip up to character	**F4**	Does not copy (skips over) the characters in the template, up to the character specified.
Quit input	**ESC**	Voids the current input; leaves the template unchanged.
Insert mode	**INS**	Enters/exits insert mode.
Replace mode	**INS F3**	Turns insert mode off; this is the default.
New template	**F5**	Makes the new line the new template.

Key **F1**

Purpose Copies one character from the template to the command line.

Comments Pressing the **F1** key copies one character from the template to the command line. When the **F1** key is pressed, one character is inserted in the command line and the insert key is disabled.

Example Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the **F1** key copies the first character (T) to the second of the two lines displayed:

```
1:*This is a sample file
F1 1:*T_
```

Each time the **F1** key is pressed, one more character appears:

```
F1 1:*Th_
F1 1:*Thi_
F1 1:*This_
```

Key **F2**

Purpose Copies multiple characters up to a given character.

Comments Pressing the **F2** key copies all characters up to a given character from the template to the command line. The given character is the next character typed after **F2**; it is not copied or displayed on the screen.

Pressing the **F2** key causes the cursor to move to the single character that is specified in the command. If the template does not contain the specified character, nothing is copied. Pressing **F2** also disables the insert key.

Example Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line.

Pressing the **F2** key copies all characters up to the character specified immediately after the **F2** key.

```
1:*This is a sample file
F2p 1:*This is a sam_
```

Key **F3**

Purpose Copies template to command line.

Comments Pressing the **F3** key copies all remaining characters from the template to the command line. Regardless of the cursor position at the time the **F3** key is pressed, the rest of the line appears, and the cursor is positioned after the last character on the line.

The insert key is disabled.

Example Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line.

Pressing the **F3** key copies all characters from the template cursor.

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

Key **DEL**

Purpose Skips over one character in the template.

Comments Pressing the **DEL** key skips over one character in the template. Each time you press the **DEL** key, one character is not copied from the template. The action of the **DEL** key is similar to the **F1** key, except that **DEL** skips a character in the template rather than copying it to the command line.

Example Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the **DEL** key skips over the first character (T).

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

The cursor position does not change and only the template is affected. To see how much of the line has been skipped over, press the **F3** key, to move the cursor beyond the last character of the line.

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

Key **F4**

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

Comments Pressing the **F4** key skips over all characters up to a given character in the template. This character is not copied and is not shown on the screen. If the template does not contain the specified character, nothing is skipped over.

The action of the **F4** key is similar to the **F2** key, except that **F4** skips over characters in the template rather than copying them to the command line.

Example Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the **F4** key skips over all the characters in the template up to the character pressed after the **F4** key:

```
1:*This is a sample file
F4p 1:*_
```

The cursor position does not change. To see how much of the line has been skipped over, press the **F3** key to copy the template. This moves the cursor beyond the last character of the line:

```
1:*This is a sample file:
F4pF3 1:*ple file._
```

Key **ESC**

Purpose Quits input and empties the command line.

Comments Pressing the **ESC** key empties the command line, but it leaves the template unchanged. **ESC** also prints a backslash (\), carriage return, and line feed, and turns insert mode off. The cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the **F3** key copies the template to the command line and the command line is identical to the original template.

Example Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Assume that you want to replace the line with "Sample File:"

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

To cancel the line you just entered (Sample File), and to keep "This is a sample file.", press **ESC**. Notice that a backslash appears on the Sample File line to tell you it has been cancelled.

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

Press **RETURN** to keep the original line, or to perform any other editing functions. If **F3** is pressed, the original template is copied to the command line:

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

Key **INS**

Purpose Enters/exits insert mode.

Comments Pressing the **INS** key causes EDLIN to enter or exit insert mode. The current cursor position in the template is not changed. The cursor moves as each character is inserted. However, when you have finished inserting characters, the cursor is positioned at the same character as it was before the insertion began. Thus, characters are inserted **in front of** the character that the cursor points to.

Example Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Assume that you press the **F2** and **f** keys:

```
1:*This is a sample file
F2f 1:*This is a sample _
```

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

1:*This is a sample file.
F2f 1:*This is a sample _
INSedit 1:*This is a sample edit _

If you now press the **F3** key, the rest of the template is copied to the line:

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

If you pressed the **RETURN** key, the remainder of the template would be truncated, and the command line would end at the end of the insert:

INSedit RETURN 1:*This is a sample edit _

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

Key **INS F3**

Purpose Enters replace mode.

Comments Pressing the **INS** and **F3** keys simultaneously causes EDLIN to exit insert mode and enter replace mode.

All the characters you type replace characters in the template. When you start to edit a line, replace mode is in effect.

If the **RETURN** key is pressed, the remainder of the template is deleted.

Example Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line.

Assume that you then press **F2**m, **INSlary, INS F3 tax, and then F3:**

```
1:*This is a sample file.
F2m 1:*This is a sa_
INSlary 1:*This is a salary_
INS F3 tax 1:*This is a salary tax_
F3 1:*This is a salary tax file._
```

Notice that you inserted **lary** and replaced **mple** with tax. If characters on the command line extend beyond the length of the template, the remaining characters in the template are automatically appended when you press **F3**.

Key **F5**

Purpose Creates a new template.

Comments Pressing the **F5** key copies the current command line to the template. The contents of the old template are deleted. Pressing **F5** outputs an @ ("at sign" character), a carriage return, and a line feed. The command line is also emptied and insert mode is turned off.

NOTE **F5** performs the same function as the **ESC** key, except that the template is changed and an @ ("at sign" character) is printed instead of a \.

Example Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line.

Assume that you enter **F2**m, **INS**lary, **INS F3** tax, and then **F3**:

```
1:*This is a sample file.
F2m 1:*This is a sa_
INSlary 1:*This is a salary_
INS F3 tax 1:*This is a salary tax_
F3 1:*This is a salary tax file._
```

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

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

The @ indicates that this new line is now the new template. Additional editing can be done using the new template.

Command Options

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

<line>

<line> indicates a line number that you type. Line numbers must be separated by a comma or a space from other line numbers, other options, and from the command.

<line> may be specified one of three ways:

Number

Any number less than 65534. If a number larger than the largest existing line number is specified, then <line> means the line after the last line number.

Period (.)

If a period is specified for <line>, then <line> means the current line number. The current line is the last line edited, and is not necessarily the last line displayed. The current line is marked on your screen by an asterisk (*) between the line number and the first character.

Pound (#).

The pound sign indicates the line after the last line number. If you specify # for <line>, this has the same effect as specifying a number larger than the last line number.

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

? The question mark option directs EDLIN to ask you if the correct string has been found. The question mark is used only with the Replace and Search commands. Before continuing, EDLIN waits for either a **Y** or **RETURN** for a yes response, or for any other key for a no response.

<string> <string> represents text to be found, to be replaced, or to replace other text. The <string> option is used only with the Search and Replace commands. Each <string> must be ended by a **CTRL Z** or a **RETURN** (see the Replace command for details). No spaces should be left between strings or between a string and its command letter, unless you want those spaces to be part of the string.

(A)ppend

Purpose

Adds the specified number of lines from disk to the file being edited in memory. The lines are added at the end of lines that are currently in memory.

Syntax

[<**n**>**A**]

Comments

This command is meaningful only if the file being edited is too large to fit into memory. As many lines as possible are read into memory for editing when you start EDLIN.

To edit the remainder of the file that does not fit into memory, lines that have already been edited must be written to disk. Then you can load unedited lines from disk into memory with the Append command.

Refer to the Write command in this appendix for information on how to write edited lines to disk.

NOTES

1. If you do not specify the number of lines to append, lines are appended to memory until available memory is 3/4 full. No action is taken if available memory is already 3/4 full.

2. The message "End of input file" is displayed when the Append command has read the last line of the file into memory.

(C)opy

Purpose

Copies a range of lines to a specified line number. The lines can be copied as many times as you want by using the <count> option.

Syntax

[<**line**>],[<**line**>],<**line**>[,<**count**>]**C**

Comments

If you do not specify a number in <count>, EDLIN copies the lines one time. If the first or the second <line> is omitted, the default is the current line. The file is renumbered automatically after the copy.

The line numbers must not overlap or an "Entry error" message appears. For example, **3,20,15C** would result in an error message.

Examples

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

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

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

1,6,7C

The result is:

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

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

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

```
6: to copy text in your file.
7: You can also use COPY
8: to copy lines of text
9: to the middle of your file.
10: End of sample file.
```

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

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

(D)elete

Purpose Deletes a specified range of lines in a file.

Syntax [<**line**>][,<**line**>]**D**

Comments If the first <line> is omitted, that option defaults to the current line (the line with the asterisk next to the line number). If the second <line> is omitted, then just the first <line> is deleted. When lines have been deleted, the line immediately after the deleted section becomes the current line and has the same line number as the first deleted <line> had before the deletion occurred.

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

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

To delete multiple lines, type

<line>,<line>D:
5,24D

The result is:

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

To delete a single line, type

6D

The result is:

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

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

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

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

```
,6D
```

The result is:

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

Notice that the lines are automatically renumbered.

<line> Edit

Purpose Edits line of text.

Syntax [<**line**>]

Comments When a line number is typed, EDLIN displays the line number and text; then, on the line below, EDLIN reprints the line number. The line is now ready for editing. You may use any of the EDLIN editing commands to edit the line. The existing text of the line serves as the template until the **RETURN** key is pressed.

WARNING If the **RETURN** key is pressed while the cursor is in the middle of the line, the remainder of the line is deleted.

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

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

To edit line 4, type:

```
4
```

The contents of the line are displayed with a cursor below the line:

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

Now, using the **F3** special editing key, type:

```
INSnumber 4: number_
F3 RETURN 4: number four.
5:*_
```

(E)nd

Purpose Ends the editing session.

Syntax **E**

Comments This command saves the edited file on disk, renames the original input file <filename>.BAK, and then exits EDLIN. If the file was created during the editing session, no .BAK file is created.

The E command takes no options. When you begin EDLIN, you must enter the filename for the output file. If the drive is not selected when EDLIN is started, the file is saved on the disk in the default drive. It is still possible to COPY the file to a different drive using the MS-DOS COPY command.

You must be sure that the disk contains enough free space for the entire file. If the disk does not contain enough free space, the write is aborted and the edited file lost, although part of the file might be written out to the disk.

Example **E RETURN**
After execution of the E command, the MS-DOS default drive prompt (for example, A>) is displayed.

(I)nsert

Purpose

Inserts text immediately before the specified <line>.

Syntax

[<**line**>]**I**

Comments

If you are creating a new file, the I command must be given before text can be typed (inserted). Text begins with line number 1. Successive line numbers appear automatically each time **RETURN** is pressed.

EDLIN remains in insert mode until **CTRL Z** is typed.

When the insert is completed and insert mode is finished, the line immediately following the inserted lines becomes the current line. All line numbers following the inserted section are incremented by the number of lines inserted.

If <line> is not specified, the default is the current line number and the lines are inserted immediately before the current line.

If <line> is any number larger than the last line number, or if a pound sign (#) is specified as <line>, the inserted lines are appended to the end of the file. In this case, the last line inserted becomes the current line.

Examples

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

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

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

```
7I
```

The result is:

```
7:_
```

Now, type the new text for line 7:

```
7: and renumber lines
```

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

```
8: CTRL     Z
```

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

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

To insert lines immediately before the current line, type:

```
I
```

The result is:

```
8: _
```

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

```
8: so they are consecutive
9: CTRL Z
```

Now to list the file and see the result, type

```
L
```

The result is:

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

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

```
10I
```

This produces the following:

```
10: _
```

Now, type the following new lines:

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

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

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: and renumber lines
8: so they are consecutive
9: in your file.
10: The insert command can place new lines
11: in the file; there's no problem
12: because the line numbers are dynamic;
13: they'll go all the way to 65533.
```

(L)ist

Purpose

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

Syntax

[<**line**>][,<**line**>]**L**

Comments

Default values are provided if either one or both of the options are omitted. If you omit the first option, 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 option.

NOTE

If the specified <line> is more than 11 lines before the current line, the display is the same as if you omitted both options. If you omit the second option, as in

<**line**>**L**

23 lines are displayed, starting with the specified <line>.

If you omit both parameters, as in

```
L
```

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

Examples

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

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

To list a range of lines without reference to the current line, type <line>,<line>L:

```
2,5L
```

The result is:

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

To list a range of lines beginning with the current line, type ,<line> L:

```
,26L
```

The result is:

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

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

```
L
```

The result is:

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

(M)ove

Purpose

Moves a range of text to the line specified.

Syntax

[<line>],[<line>],<line>M

Comments

Use the Move command to move a block of text (from the first <line> to the second <line>) to another location in the file.

Text is inserted before the target line.

The lines are renumbered according to the direction of the move. For example,

```
,+25,100M
```

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

(P)age

Purpose Pages through a file 23 lines at a time.

Syntax [<**line**>][,<**line**>]**P**

Comments If the first <line> is omitted, that number defaults to the current line plus one. If the second <line> is omitted, 23 lines are listed. The new current line becomes the last line displayed and is marked with an asterisk.

(Q)uit

Purpose

Quits the editing session, does not save **any** editing changes, and exits to the MS-DOS operating system.

Syntax

Q

Comments

EDLIN prompts you to make sure you don't want to save the changes.

Type **Y** if you want to quit the editing session.

No editing changes are saved and no .BAK file is created. Refer to the End command in this Appendix for information about the .BAK file.

Type **N** or any other character except Y if you want to continue the editing session.

NOTE

When started, EDLIN erases any previous copy of the file with an extension of .BAK to make room to save the new copy. If you reply **Y** to the Abort edit (Y/N)? message, your previous backup copy no longer exists.

Example

```
Q
Abort edit (Y/N)?Y      RETURN
A>_
```

(R)eplace

Purpose

Replaces all occurrences of a string of text in the specified range with a different string of text or blanks.

Syntax

[<**line**>][,<**line**>][**?**]**R**<**string1**> **CTRL Z** <**string2**>

Comments

As each occurrence of <string1> is found, it is replaced by <string2>. Each line in which a replacement occurs is displayed. If a line contains two or more replacements of <string1> with <string2>, then the line is displayed once for each occurrence.

When all occurrences of <string1> in the specified range are replaced by <string2>, the R command terminates and the asterisk prompt reappears.

If a second string is to be given as a replacement, then <string1> must be separated from <string2> with a **CTRL Z**. <String2> must end with a **CTRL Z RETURN** combination or with a simple **RETURN**.

If <string1> is omitted, then Replace takes the old <string1> as its value. If there is no old <string1>, i.e., this is the first replace done, then the replacement process ends immediately.

If the first <line> is omitted in the range argument (as in ,<line>) then the first <line> defaults to the line **after** the current line. If the second <line> is omitted (as in <line> or <line>,), the second <line> defaults to #. Therefore, R<string1> <string2> is the same as <line> + 1,#. Remember that # indicates the line after the last line of text.

If <string1> is ended with a **CTRL Z** and there is no <string2>, <string2> is taken as an empty string and becomes the new replace string. For example,

R<string1> **CTRL Z RETURN**

deletes all of <string1>, but

R<string1> **RETURN**

and

R **RETURN**

both use the old <string2> and the later example also uses the old <string1>. Note that "old" here refers to a previous string specified either in a Search or a Replace command.

If the question mark (?) option is used, the Replace command stops at each line with a string that matches <string1>, displays the line with <string2> in place, and then displays the prompt O.K.?. If you press **Y** or the **RETURN** key, then <string2> replaces <string1>, and the next occurrence of <string1> is found.

Again, the O.K.? prompt is displayed. This process continues until the end of the range or until the end of the file. After the last occurrence of <string1> is found, EDLIN displays the asterisk prompt.

If you press any key besides **Y** or **RETURN** after the O.K.? prompt, <string1> is left as it was in the line, and Replace goes to the next occurrence of <string1>. If <string1> occurs more than once in a line, each occurrence of <string1> is replaced individually, and the O.K.? prompt is displayed after each replacement. In this way, only the desired <string1> is replaced, and you can prevent unwanted substitutions.

Examples

Assume that the following file exists and is ready for editing:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.
8: The insert command can place new lines
9: in the file; there's no problem
10: because the line numbers are dynamic;
11: they'll go all the way to 65533.
*
```

To replace all occurrences of <string1> with <string2> in a specified range, type:

2,12 Rand CTRL Z or RETURN

The result is:

```
4: Delete or Insert
5: (the D or I commors)
8: The insert commors can place new lines
```

Note that in the above replacement, some unwanted substitutions have occurred. To avoid these and to confirm each replacement, the same original file can be used with a slightly different command.

In the next example, to replace only certain occurrences of the first <string> with the second <string>, type:

2? Rand CTRL Z or RETURN

The result is:

```
4: Delete or Insert
   O.K.? Y
5: (The D or I commands)
   O.K.? Y
5: (The D or I commors)
   O.K.? N
8: The insert commor can place new lines
   O.K.? N
*_
```

Now, type the List command **(L)** to see the result of all these changes:

```
.
.
4: Delete or Insert
5: (The D or I commands)
.
8: The insert command can place new lines
```

(S)earch

Purpose

Searches the specified range of lines for a specified string of text.

Syntax

[<**line**>][,<**line**>][**?**]**S**<**string**> **RETURN**

Comments

The <string> must be ended with a **RETURN**. The first line that matches <string> is displayed and becomes the current line. If the question mark option is not specified, the Search command ends when a match is found. If no line contains a match for <string>, the message "Not found" is displayed.

If the question mark option (?) is included in the command, EDLIN displays the first line with a matching string; it then prompts you with the message O.K.?. If you press either the **Y** or **RETURN** key, the line becomes the current line and the search ends.

If you press any other key, the search continues until another match is found, or until all lines have been searched (and the Not found message is displayed). If the first <line> is omitted (as in first <line> defaults to the line **after** the current line. If the second <line> is omitted (as in <line> S<string> or <line>, S<string>), the second <line> defaults to # (line after last line of file).

This is the same as typing <line>, #S <string>.

If <string> is omitted, Search takes the old string if there is one. (Note that "old" here refers to a string specified in a previous Search or Replace command.) If there is not an old string (i.e., no previous search or replace has been done), the command ends immediately.

Examples

Assume that the following file exists and is ready for editing:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.
8: The insert command can place new lines
9: in the file; there's no problem
10: because the line numbers are dynamic;
11:*they'll go all the way to 65533.
```

To search for the first occurrence of the string "and", type

1,12 Sand RETURN

The following line is displayed:

```
4: Delete and Insert
```

To get the "and" in line 5, modify the search command by typing:

DEL F3 ,12 Sand RETURN

The search then continues from the line after the current line (line 4), since no first line was given. The result is:

```
5: (the D and I commands)
```

To search through several occurrences of a string until the correct string is found, type:

1, ? Sand

The result is:

```
4: Delete and Insert
O.K.?_
```

If you press any key (except **Y** or **RETURN**), the search continues, so type N here:

```
O.K.? N
```

Continue:

```
5: (the D and I commands)
   O.K.?_
```

Now press **Y** to terminate the search:

```
O.K.? Y
*_
```

To search for string XYZ from the line after the current line to the end without the verification (O.K.?), type:

```
SXYZ
```

EDLIN reports a match and continues to search for the same string when you issue the S command:

```
S
```

Transfer

Purpose

Inserts (merges) the contents of <filename> into the file currently being edited at <line>. If <line> is omitted, then the current line is used.

Syntax

[<**line**>]**T**<**filename**>

Comments

This command is useful if you want to put the contents of a file into another file or into the text you are typing. The transferred text is inserted at the line number specified by <line> and the lines are renumbered.

(W)rite

Purpose

Writes a specified number of lines to disk from the lines that are being edited in memory. Lines are written to disk beginning with line number 1.

Syntax

[<**n**>]**W**

Comments

This command is meaningful only if the file you are editing is too large to fit into memory. When you start EDLIN, EDLIN reads lines into memory until memory is 3/4 full.

To edit the remainder of your file, you must write edited lines in memory to disk. Then you can load additional unedited lines from disk into memory by using the Append command.

NOTE

If you do not specify the number of lines, lines are written until memory is 3/4 full. No action is taken if available memory is already less than 3/4 full. All lines are renumbered, so that the first remaining line becomes line number 1.

Error Messages

When EDLIN finds an error, one of the following error messages is displayed:

Cannot edit .BAK file—rename file

You attempted to edit a file with a filename extension of .BAK. .BAK files cannot be edited because this extension is reserved for backup copies.

If you need the .BAK file for editing purposes, you must either RENAME the file with a different extension; or COPY the .BAK file and give it a different filename extension.

No room in directory for file

When you attempted to create a new file, either the file directory was full or you specified an illegal disk drive or an illegal filename.

Check the command line that started EDLIN for illegal filename and illegal disk drive entries. If the command is no longer on the screen and if you have not yet typed a new command, the EDLIN start command can be recovered by pressing the **F3** key. If this command line contains no illegal entries, run the CHKDSK program for the specified disk drive. If the status report shows that the disk directory is full, remove the disk. Insert and format a new disk.

Entry Error

The last command typed contained a syntax error.

Retype the command with the correct syntax and press **RETURN**.

Line too long

During a Replace command, the string given as the replacement caused the line to expand beyond the limit of 253 characters. EDLIN aborted the Replace command.

Divide the long line into two lines, then try the Replace command.

Disk Full—file write not completed

You gave the End command, but the disk did not contain enough free space for the whole file. EDLIN aborted the E command and returned you to the operating system. Some of the file may have been written to the disk.

Only a portion (if any) of the file has been saved. You should probably delete that portion of the file and restart the editing session. The file is not available after this error. Always be sure that the disk has sufficient free space for the file to be written to disk **before** you begin your editing session.

Incorrect DOS version

You attempted to run EDLIN under a version of MS-DOS that was not 2.0 or higher.

You must make sure that the version of MS-DOS that you are using is 2.0 or higher.

Invalid drive name or file

You have not specified a valid drive or filename when starting EDLIN.

Specify the correct drive or filename.

Filename must be specified

You did not specify a filename when you started EDLIN.

Specify a filename.

Insufficient memory

There is not enough memory to run EDLIN.

You must free some memory by saving the contents of memory to disk or by clearing memory before restarting EDLIN.

File not found

The filename specified during a Transfer command was not found.

Specify a valid filename when issuing a Transfer command.

Must specify destination number

A destination line number was not specified for a Copy or Move command.

Reissue the command with a destination line number.

Not enough room to merge the entire file

There was not enough room in memory to hold the file during a Transfer command.

You must free some memory by writing some files to disk or by deleting some files before you can transfer this file.

File creation error

The EDLIN temporary file cannot be created.

Check to make sure that the directory has enough space to create the temporary file. Also, make sure that the file does not have the same name as a subdirectory in the directory where the file to be edited is located.

C LINK

- Overview
- LINK File Usage
- Segments, Groups, and Classes
- Invoking LINK
- Sample LINK Session
- LINK Error Messages

Overview

LINK is an executable program on your MS-DOS Supplemental Programs diskette. LINK combines object modules that are the output of the MACRO-86 assembler or a compatible compiler. It produces a relocatable run file (load module) and a list of external references and error messages.

To run LINK, you provide object, run, list, and library file parameters. You may optionally enter switches that modify the operation of LINK.

"Invoking the Linker" describes the three ways to run LINK: interactive entry, command line entry, and automatic response file entry. Interactive entry is used most frequently, so its section contains information common to all three methods.

If you are linking a high-level language program, the compiler determines the arrangement of your object modules in memory. If you are using assembler, however, you have more control over your program's organization. The section "Segments, Groups, and Classes" shows you how to specify the order of your object modules at run time.

LINK File Usage

The link process involves the use of several files.

LINK:

- Works with one or more input files
- Produces two output files
- Creates a temporary disk file if necessary
- Searches up to eight library files

The format for LINK filenames is the same as that of any disk file:

Syntax [d:][pathname]filename[.ext]

d: the drive designation. Permissible drive designations for LINK are A: through O:.

path a path of directory names.

filename any legal filename of one to eight characters.

ext a one- to three-character extension to the filename.

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

.OBJ Object
.LIB Library

LINK appends the following default extensions to the output (Run and List) files:

.EXE Run (may not be overridden)
.MAP List (may be overridden)

VM.TMP File

LINK uses available memory for the link session. If an output file exceeds available memory, LINK creates a temporary file, names it VM.TMP, and puts it on the disk in the default drive. If LINK creates VM.TMP, it will display the message:

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

Once this message is displayed, do not remove the diskette from the default drive until the link session ends. If the diskette is removed, the operation of LINK is unpredictable and LINK usually displays the error message:

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

LINK writes the contents of VM.TMP to the file named following the Run File: prompt. VM.TMP is a working file only and is deleted at the end of the linking session.

Do not use VM.TMP as a filename for any file. If LINK requires the VM.TMP file, LINK deletes the VM.TMP already on disk and creates a new VM.TMP. Thus, the contents of the previous VM.TMP file are lost.

Changing diskettes

You may want to change diskettes during the link operation. If LINK cannot find an object file on the specified diskette, it prompts you to change diskettes rather than aborting the session. If you enter the /PAUSE switch, LINK pauses and prompts you to change diskettes before it creates the run file. You may change diskettes when prompted except in the following cases:

- the diskette you want to change has a VM.TMP file on it.
- you have requested a list file on the diskette you want to change.

Segments, Groups, and Classes

Below terms are explained to help you understand how LINK works. Generally, if you are linking object modules from a high-level language compiler, you do not need to know these terms. If you are linking assembly language modules, read this section carefully.

Segment

The segment is one of the most basic units of program memory organization. A segment is a contiguous area of memory up to 64K bytes long, and may be located anywhere in RAM. The contents of a segment are addressed by a segment:offset address pair, where "segment" is the segment's base or lowest address (see "The 20-Bit Address" in chapter 4).

Each segment has a class name in addition to its segment name. All segments with the same class name are loaded into memory contiguously by the linker from the first segment of that class to the last.

Class

A class is a collection of related segments. By naming the segments of your assembly language program to classes, you control the order in which they are loaded into memory (for high level languages, the compiler does this for you).

LINK loads segments into memory on a class-by-class basis. Starting with the first class encountered in the first object file, all of the segments of each class are loaded. Within each class, the linker loads the segments in the order in which it finds them in the object files. Therefore, you can control the order in which classes are loaded by the order in which segments from different classes appear in the object files.

To ensure that classes are loaded in the order you desire, you can create a dummy module to feed to the linker as the first object file. This module declares empty-segment classes in the order you want the classes loaded. For example, one such file might look like this:

```
A  SEGMENT 'CODE'
A  ENDS
B  SEGMENT 'CONST'
B  ENDS
C  SEGMENT 'DATA'
C  ENDS
D  SEGMENT STACK 'STACK'
D  ENDS
```

If this method is used, be sure to declare all the classes used in your program in the dummy module; otherwise, you lose control over the ordering of classes. Also, this method should only be used when linking assembly language programs. Do not create a dummy module if linking object files for a compiler, or unpredictable results may occur. Classes may be any length.

Group

Just as classes allow you to combine segments in a way that is logical, groups combine segments in 64K byte chunks to make them easily addressable. The segments in a group need not be contiguous, but when loaded they must fit within 64K bytes. This way each segment in the group can be fully addressed by an offset to one segment address, which is the start address of the lowest segment in the group. Segments are named to groups by the assembler or compiler or, as is possible in assembly language programs, by the programmer. Note that a segment can be large enough to be an entire group by itself.

Invoking LINK

Ways to Invoke LINK

LINK is invoked in one of three ways. The first method, interactive entry, requires you to respond to individual prompts.

For the second method, command line entry, type all commands on the same line used to start LINK.

To use the third method, automatic response file entry, create a response file that contains all the necessary commands and tell LINK where that file is when you run LINK.

Interactive Entry	LINK
Command Line Entry	LINK filenames[/switches]
Automatic Response File Entry	LINK @filespec

Interactive Entry

To invoke LINK interactively, type:

```
LINK
```

LINK loads into memory, then displays four prompts, one at a time. At the end of each line, after typing your response to the prompt, you may type one or more switches preceded by a forward slash.

The command prompts are summarized below. Defaults appear in square brackets ([]) after the prompt. **Object Modules** is the only prompt that requires a response.

LINK Prompts

Prompt	Responses
Object Modules(.OBJ):	(d:)(pathname)filename(.ext) (+((d:)(pathname)filename(.ext))...)
Run File(filename.EXT):	(d:)(pathname)(filename(.ext))
List File(NUL.MAP):	(d:)(pathname)(filename(.ext))
Libraries(.LIB):	(d:)(pathname)(filename(.ext)) (+((d:)(pathname)filename(.ext))...)

Notes:

- If you enter a filename without specifying the drive, the default drive is assumed. If you enter a filename without specifying the pathname, the default path is assumed. The libraries prompt is an exception — if the linker looks for the libraries on the default drive and doesn't find them, it looks on the drive specified by the compiler.

- To select default responses to all remaining prompts, use a single semicolon (;) followed immediately by <return> at any time after the second prompt (Run File:).

  Once you enter the semicolon, you can no longer respond to any of the prompts for that link session. Use the <RETURN> key to skip prompts.

- Use <CONTROL-C> to abort the link session at any time.

Object Modules to be Included

Object Modules (.OBJ):

List .OBJ files to be linked. They must be separated by blank spaces or plus signs (+). If the plus sign is the last character typed, this prompt will reappear so that you can enter more object modules.

LINK assumes that object modules have the extension .OBJ unless you explicitly specify some other extension. Object filenames may not begin with the @ symbol (@ is used for specifying an automatic response file).

The order in which you key in the object files is significant. See section on segments, groups, and classes for more information.

Load Module

Run File (Obj-file.EXE):

Give filename for executable object code. The default is:<first-object-filename>.EXE. (You cannot change the output extension.) You can specify just the drive designation or just a pathname for this prompt.

Listing

List File (NUL.MAP):

Give filename for listing (also known as a linker map). The listing is not created if you select the default. You can request a listing by entering a drive designator, pathname, or filename[.ext]. If you do not specify an extension, the default .MAP is used.

You can have the listing printed by specifying a print device instead of a filename or have the listing displayed on the screen by specifying CON. If you display the linker map, you can also print it by pressing Ctrl-PrtSc.

Libraries to be Searched

Libraries (.LIB):

List filenames to be searched separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear.

LINK searches library files in the order listed to resolve external references. When it finds the module that defines the external symbol, LINK processes that module as another object module.

There is no default library search for MACRO assembler object modules. For compiled modules, if you select the default for this prompt, LINK looks for the compiler package's library on the default drive. If not found there, LINK looks on the drive specified by the compiler.

If LINK cannot find a library file, it displays:

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

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

If two libraries have the same filename, only the first in the list is searched.

LINK Switches

The seven LINK switches control various LINK functions. Type switches at the end of a prompt response regardless of which method you use to start LINK. Switches may be grouped at the end of any response, or may be scattered at the end of several. Even if you type more than one switch at the end of one response, each switch must be preceded by a forward slash (/).

All switches may be abbreviated. The only restriction is that an abbreviation must be sequential from the first letter through the last typed; no gaps or transpositions are allowed. For example:

Legal	Illegal
/D	**/DSL**
/DS	**/DAL**
/DSA	**/DLC**
/DSALLOCA	**/DSALLOCT**

/DSALLOCATE

/DSALLOCATE tells LINK to load all data at the high end of the Data Segment. Otherwise, LINK loads all data at the low end of the Data Segment. At runtime, the DS pointer is set to the lowest possible address to allow the entire DS segment to be used. Use of /DSALLOCATE in combination with the default load low (that is, the /HIGH switch is not used) permits the user application to dynamically allocate any available memory below the area specifically allocated within DGroup yet to remain addressable by the same DS pointer. This dynamic allocation is needed for Pascal and FORTRAN programs.

Your application program may dynamically allocate up to 64K bytes (or the actual amount of memory available) less the amount allocated within DGroup.

/HIGH

/HIGH causes LINK to place the Run file as high as possible in memory. Otherwise, LINK places the Run file as low as possible.

Note:
Do not use /HIGH with Pascal or FORTRAN programs.

/LINENUMBERS

/LINENUMBERS tells LINK to include in the List file the line numbers and addresses of the source statements in the input modules. Otherwise, line numbers are not included in the List file.

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

/MAP

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

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

/PAUSE

/PAUSE causes LINK to pause in the link session when the switch is encountered. Normally, LINK performs the linking session from beginning to end without stopping. This switch enables you to swap the diskettes before LINK outputs the Run (.EXE) file.

When LINK encounters /PAUSE, it displays the message:

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

LINK resumes processing when you press any key.

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

/STACK: <number>

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

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

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

```
WARNING: NO STACK STATEMENT
```

/NO

/NO is short for NODEFAULTLIBRARY-SEARCH. This switch applies only to higher level language modules. This switch tells LINK not to search the default libraries in the object modules. For example, if you are linking object modules in Pascal, specifying /NO tells LINK not to automatically search the library named PASCAL.LIB to resolve external references.

Command Line Entry

Purpose

You may invoke LINK by typing all commands on one line. The entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas. Use the following syntax:

Syntax

LINK <obj-list>,<runfile>,<listfile>,<lib-list>[/switch...]

obj-list

a list of object modules, separated by plus signs or spaces.

runfile

name of the file to receive the executable output.

listfile

name of the file to receive the listing.

lib-list

list of library modules to be searched, separated by spaces or plus signs.

/switch

refers to optional switches which may be placed following any of the response entries (just before any of the commas or after the <lib-list>, as shown).

To select the default for a field, simply type a second comma with no spaces between the two commas.

Example:

```
LINK FUN + TEXT + TABLE + CARE,
FUNLIST, COBLIB.LIB
```

This command causes LINK to load. Then the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ and CARE.OBJ are loaded. LINK links the object modules and writes the output to FUN.EXE (by default), creates a List file named FUNLIST.MAP, and searches the library file COBLIB.LIB.

Automatic Response File Entry

It is often convenient to save responses to the linker for re-use at a later time. This is especially useful when a long list of object modules needs to be specified. The use of an automatic response file allows you to do this.

Before using this option, you must create the response file. Each line of text corresponds to one LINK prompt. The responses must be typed in the same order as they are when entered interactively. To continue a line, type a plus sign (+) at the end of the line.

You can enter the name of more than one automatic response file on the command line and combine response file names with additional parameters. The combined series of resulting parameters must be a valid sequence of LINK prompts.

Use switches and special characters (+ and ;) in the response file the same way they are used when entered interactively.

To invoke the linker using a response file, type

LINK @ <filename>

Filename is the name of a response file.

When the session begins, LINK displays each prompt with the corresponding response from the response file. If the response file does not contain answers for all the prompts, LINK displays the prompt which does not have a response and waits for a response. When you type a legal response, LINK continues the link session.

Example:

```
FUN TEXT TABLE CARE
/PAUSE /MAP
FUNLIST
COBLIB.LIB
```

This response file tells LINK to load the four object modules named FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ. LINK pauses before producing a public symbol map to permit you to swap disks. When you press any key, the output files will be named FUN.EXE and FUN-LIST.MAP. LINK will search the library file COBLIB.LIB.

Sample LINK Session

This sample shows you the type of information displayed during an LINK session.

In response to the MS-DOS prompt, type:

```
LINK
```

The system displays the following messages and prompts:

```
    Microsoft Object Linker V2.01 (Large)
© Copyright 1982, 1983 by Microsoft Inc.

Object Modules (.OBJ): IO SYSINIT
Run File (IO.EXE):
List File (NUL.MAP): PRN /MAP /LINE
Libraries (.LIB): ;
```

Notes:

- By specifying /MAP, you get both an alphabetic listing and a chronological listing of public symbols.
- By responding PRN to the List File: prompt, you can redirect your output to the printer.
- By specifying the /LINE switch, LINK gives you a listing of all line numbers for all modules. (Note that /LINE can generate a large volume of output.)

Once LINK locates all libraries, the linker map displays a list of segments in the order of their appearance within the load module. The list might look like this:

```
Start     Stop      Length    Name
00000H    009ECH    09EDH     CODE
009F0H    01166H    0777H     SYSINITSEG
```

The information in the Start and Stop columns shows the 20-bit hex address of each segment relative to location zero. Location zero is the beginning of the load module.

The addresses displayed are not the absolute addresses where these segments are loaded. See the following section on the DEBUG program for information on how to determine the absolute address of a segment.

Because the /MAP switch was used, LINK displays the public symbols by name and value. For example:

```
ADDRESS         PUBLICS BY NAME
009F:0012       BUFFERS
009F:0005       CURRENT DOS LOCATION
009F:0011       DEFAULT DRIVE
009F:000B       DEVICE LIST
009F:0013       FILES
009F:0009       FINAL DOS LOCATION
009F:000F       MEMORY SIZE
009F:0000       SYSINIT

ADDRESS         PUBLICS BY VALUE
009F:0000       SYSINIT
009F:0005       CURRENT DOS LOCATION
009F:0009       FINAL DOS LOCATION
009F:000B       DEVICE LIST
009F:000F       MEMORY SIZE
009F:0011       DEFAULT DRIVE
009F:0012       BUFFERS
009F:0013       FILES
```

The final line in the listing file describes the program's entry point:

```
Program entry point at 0009F:0000
```

LINK Error Messages

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

Attempt to access data outside of segment bounds, possibly bad object module

There is probably a bad object file.

Bad numeric parameter

Numeric value is not in digits.

Cannot open temporary file

LINK is unable to create the file VM.TMP because the disk directory is full. Insert a new disk. Do not remove the disk that will receive the List. MAP file.

Error: dup record too complex

DUP record in assembly language module is too complex. Simplify DUP record in assembly language program.

Error: fixup offset exceeds field width

An assembly language instruction references an address with a short or near instruction instead of a long or far instruction. Edit assembly language source and reassemble.

Input file read error

There is probably a bad object file.

Invalid object module

An object module(s) is incorrectly formed or incomplete (as when assembly is stopped in the middle).

Symbol defined more than once

LINK found two or more modules that define a single symbol name.

Program size or number of segments exceeds capacity of linker

The total size may not exceed 384K bytes and the number of segments may not exceed 255.

Requested stack size exceeds 64K

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

Segment size exceeds 64K

64K bytes is the addressing system limit.

Symbol table capacity exceeded

Very many and/or very long names were typed exceeding the limit of approximately 50K bytes.

Too many external symbols in one module

The limit is 256 external symbols per module.

Too many groups

The limit is ten groups.

Too many libraries specified

The limit is 8 libraries.

Too many public symbols

The limit is 1024 public symbols.

Too many segments or classes

The limit is 256 (segments and classes together must total 256 or less).

Unresolved externals: <list>

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

VM read error

This is a disk error; it is not caused by LINK.

Warning: no stack segment

None of the object modules specified contains a statement allocating stack space.

Warning: segment of absolute or unknown type

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

Write error in TMP file

No more disk space remains to expand the VM.TMP file.

Write error on run file

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

D Debug

- Overview
- How to Invoke DEBUG
- Debugging Commands
- Command Parameters

  A Assemble
  C Compare
  D Display
  E Enter
  F Fill
  G Go
  H Hexarithmetic
  I Input
  L Load
  M Move
  N Name
  O Output
  Q Quit
  R Register
  S Search
  T Trace
  U Unassemble
  W Write

- DEBUG Error Messages

Overview

The DEBUG utility is an executable object program that resides on your MS-DOS Supplemental Programs diskette. DEBUG performs the following functions:

- Allows you to single step through a program, instruction by instruction, for testing purposes.
- Changes register and file contents during the DEBUG session so that you can test a code change without reassembling your program.
- Makes permanent changes to diskette files so you can use DEBUG to recover files that may otherwise be lost.
- Supports a disassemble command so you can translate machine code instructions into their assembly language equivalents for testing purposes.

How to Invoke DEBUG

The DEBUG program is invoked as follows:

```
DEBUG [filename [.arglist]]
```

filename the name of the program file to be debugged.

arglist An optional list of filename parameters and switches. These will be passed to the program specified by the filename parameter. When the program is loaded into memory, it is loaded as if it had been invoked with the command

```
filename arglist
```

That is, filename indicates the file to be debugged, and arglist is the rest of the command line that is used when the file is invoked and loaded in memory via COMMAND.COM.

If you enter DEBUG without parameters, since no filename has been specified, current memory, disk blocks, or disk files can be manipulated.

Comments

On entering the DEBUG environment DEBUG responds with the hyphen (-) prompt and underline (_) cursor. You now may enter any DEBUG command.

If you include the filename in the command line, the specified file is loaded into memory starting at location 100 (hexadecimal). However, if you specify a file with a .EXE extension, the program is relocated to the address specified in the header of the file. See the chapter on "Program Structure and Loading" for information on the format of the file header.

If the file has the HEX extension, the file is loaded beginning at the address specified in the HEX file. HEX files are in INTEL hex format and are converted to memory image format by DEBUG.

All DEBUG commands may be aborted at any time by pressing <CTRL-C>. Pressing <CTRL-S> suspends the display, so that you can read it before the output scrolls away. After suspending the display, press any key (except <CTRL-S> or <CTRL-C>) to continue scrolling.

Examples

```
DEBUG <CR>.
```

The DEBUG session begins, but without loading a file.

```
DEBUG b:myprog <CR>.
```

The DEBUG environment is entered and the file named "myprog" is loaded into memory from drive B.

When you invoke DEBUG, it sets up a program segment prefix at offset 0 in the program work area. You can overwrite this area if you enter DEBUG without parameters. Moreover, if you are debugging a file with a COM or EXE extension, do not tamper with the program header below location 5CH, or DEBUG will terminate.

Do not restart a program after a "Program terminated normally" message is displayed. You must reload the program with the N and L commands for it to run properly.

Debugging Commands

This section describes the DEBUG commands in alphabetical order for ease of reference.

- Commands can be entered in either upper or lower case.
- Command keywords and command parameters can be separated from each other by spaces or commas for readability but need not be, except where two hexadecimal numbers are entered as parameters, in which case they must be separated by a comma or space. For brevity, the syntax of this chapter will always indicate a comma where separation is obligatory, but note that a space can alternatively be used.
- Commands only become effective after entering <CR>.
- If you make a syntax error when entering a command, the message "Error" will be displayed. You must re-enter the command using the correct syntax.

Command Parameters

The following DEBUG command parameters require definition.

address a hex value in one of the following formats:

- a segment register designation and a hex offset separated from each other by a colon. For example:

  DS:0300

- a hexadecimal segment and offset separated from each other by a colon. For example:

  9DO:0100

- a hexadecimal offset value. The DEBUG command will use a default segment value from either the DS or CS registers, depending on the command. For example:

  200

byte a one or two character hexadecimal value.

drive 0, 1, or 2 depending on whether you wish to select drive A, drive B or drive C, respectively.

range a range of addresses. The range can be specified as

address L value

where address specifies the start of the range and value specifies the length of the range. For example:

DS:300L30

indicates a range of 48 locations starting at address 300 in the segment indicated by the DS register.

The specified range cannot be greater than 10000 (hexadecimal). To specify this value enter 0000 (or 0) as the value parameter.

A range can also be specified as:

address,address

where the two addresses indicate the limits of the range. A space may be used instead of a comma.

value a 1 to 4 character hexadecimal value.

A (ASSEMBLE)

Assembles 8086 mnemonics directly into memory.

Syntax

A [address]

Address is the start address into which the subsequently entered line of mnemonics is to be assembled. If this parameter is omitted, offset 100 from the segment in the CS register is assumed, if you did not enter an Assemble command previously. If you did enter Assemble previously, the code assembles into the address following the last instruction loaded by the previous Assemble command.

Comments

- After you enter the Assemble command, DEBUG displays the specified address followed by the cursor. You may then enter a line of 8086 assembler mnemonics. On terminating the line with <CR>, the line will be assembled into memory starting at the specified location. The address of the byte subsequent to the assembled code will be displayed on the next line along with the cursor to enable you to enter the next line of code. If, instead of a line of 8086 mnemonics, you simply enter <CR>, the Assemble command terminates and the DEBUG prompt reappears.

- All numeric values are hexadecimal and must be entered as 1 to 4 characters without a trailing H. Prefix mnemonics must be specified in front of the opcode to which they refer. You may also enter them on a separate line.

- The segment override mnemonics are CS:, DS:, ES: and SS:. The mnemonic for the far return is RETF. String manipulation mnemonics must explicitly state the string size. For example, use MOVSB to move byte strings.

- The Assemble command will automatically assemble short, near, or far jumps and calls, depending on byte displacement with respect to the destination address. These may be overridden with the NEAR or FAR prefix. For example:

```
0100:0500 JMP 502          ;a two-byte
                           ;short jump
0100:0502 JMP NEAR 505     ;a three-byte
                           ;near jump
0100:505 JMP FAR 50A       ;a five-byte far
                           ;jump
```

  The NEAR prefix may be abbreviated to NE, but the FAR prefix cannot be abbreviated.

- DEBUG cannot tell whether some operands refer to a word memory location or to a byte memory location. In this case the data type must be explicitly stated with the prefix "WORD PTR" or "BYTE PTR". Acceptable abbreviations are "WO" and "BY". For example:

```
NEG     BYTE PTR [128]
DEC     WO [SI]
```

- DEBUG cannot distinguish whether an operand refers to a memory location or to an immediate operand. Enclose operands that refer to memory locations in square brackets. For example:

```
MOV AX,21          ;Load AX with 21H
MOV AX,[21]        ;Load AX with the contents of
                   ;location 21H
```

- Two pseudo-instructions are available with the Assemble command. The DB opcode will assemble byte values directly into memory. The DW opcode assembles word values 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"
```

- The Assemble command supports all forms of register indirect addressing. 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
```

Example

1. Enter **A200<CR>.**
2. DEBUG displays 09AC:0200_.
3. Enter **MOV AX,(21)<CR>.**
4. The 8086 mnemonics are assembled starting at location 200. The byte location subsequent to the assembled code is then displayed:

   09AC:0203_

5. Enter **<CR>.**
6. The Assemble command terminates and the DEBUG prompt reappears.

C (COMPARE)

Compares the contents of two areas of memory.

Syntax

C range,address

range

the range of addresses defining the first area to be compared. If no segment is specified, then the segment specified in the DS register is assumed.

address

the start of the area to be compared with the area specified by the range parameter.

Comments

- The Compare command compares the area of memory specified by the range parameter with an area of the same size starting at the location specified by the address parameter.
- If the contents of the two areas are identical, nothing is displayed. If there are differences, then the differences are displayed in the form

  <address1> <contents1> <contents2> <address2>

  <address1> indicates the address in the first area and <contents1> its contents. <address2> indicates the corresponding address in the second area and <contents2> its contents.

Example

1. Enter **C100,1FF,300<CR>** or **C100L100, 300<CR>.**
2. The area of memory from 100 to 1FF is compared with the area of memory from 300 to 3FF.

D (DISPLAY)

Displays an area of memory.

Syntax

D [range] or
D [address]

range the range of addresses whose contents are to be displayed. If you enter only offsets, then the segment specified in the DS register is assumed.

address the address from which the display is to start. The contents of this address and the subsequent 127 locations are displayed. If only an offset is entered, then the segment specified in the DS register is assumed.

Comments

- If D is specified without parameters, then the 128 bytes following the last address to be displayed are displayed. If no location has yet been accessed, the display will start from location DS:100.
- If D and the range parameter are specified, the contents of that range of addresses are displayed. If this takes more than 24 screen lines, the display is scrolled until the contents of the final address in the range are displayed on line 24.
- The display is displayed in two portions:

  A hexadecimal display, where each byte is represented by its hexadecimal value, and an ASCII display, where the equivalent ASCII character for the byte is displayed. If there is no corresponding printable ASCII character, a period (.) is displayed.

- Each line of the display begins with an address followed by the hexadecimal contents of the 16 bytes starting from the addressed location. The eighth and ninth bytes are separated by a hyphen (-). The right-hand columns display the equivalent ASCII values. Each line of the display, except possibly the first, begins on a 16 byte boundary.

Example

1. Enter **D 100,110<CR>**
2. Lines 100H to 110H (inclusive) are displayed.
3. Enter **D<CR>.**
4. The 128 bytes starting from location 111H are displayed.
5. Enter **D200<CR>.**
6. The 128 bytes starting from location 200H are displayed.

E (ENTER)

Replaces the contents of memory locations at the byte address(es) specified.

Syntax

E address[,bytevalue[,bytevalue...]]

address

the address of the location whose value is to be replaced; or the address of the first of a succession of locations whose contents are to be replaced. If only an offset is specified, then the segment indicated by the DS register is assumed.

bytevalue

the value that is to replace the contents of the specified address. The first bytevalue parameter will replace the contents of the location specified by the address parameter. A second bytevalue will replace the contents of the location following that specified by the address parameter, and so on.

Comments

- If the command is entered without the byte value list, then DEBUG displays the specified address and its contents. The Enter command then waits for you to perform one of the following:

  1 Replace the displayed bytevalue by entering another value. Enter the new value after the current value. If you enter an illegal value, or if you type more than two digits, the illegal or extra character is not echoed.

2 Advance to the next byte by pressing <SPACE>. To change the value of this byte simply enter the value as described above. If you advance beyond an eight-byte boundary, DEBUG starts a new display line with the address displayed at the start of the line. To advance to the next byte without changing the current byte, press <SPACE> again.

3 To return to the previous byte enter hyphen (-). DEBUG then starts a new display line with the address of the byte you have returned to and its contents. You can then change the contents of this location as described above. To move back one byte further without changing this value, enter hyphen again, and another new display line will be generated.

4 Terminate the Enter command by pressing <CR>. This key may be pressed in any byte position.

- If you specify byte values in the command line, then the first of these byte values will replace the contents of the location specified by the address parameter. Subsequent entries in the list of byte values will replace subsequent bytes in memory.

Example

1 Enter **E 100<CR>**

2 DEBUG displays something like 058D:0100 CD._

3 Enter **26.**

4 the value of location 100 is changed to 26 and DEBUG displays:

058D:0100 CD.26_

5 Enter **<SPACE>.**

6 The next byte (location 101) is displayed

```
058D:0100 CD.26 20._
```

7 Enter **<SPACE>.**

8 The next byte (location 102) is displayed

```
058D:100 CD.26 20. 00._
```

9 Enter **<->.**

10 The previous byte (location 101) is displayed on the next line

```
058D:0100 CD.26 20. 00.
058D:0101 20._
```

11 Enter **30<CR>.**

12 The contents of location 101 are changed to 30 and the Enter command is terminated.

```
058D:0100 CD.26. 20. 00.
058D:0101 20.30
```

13 Enter **E 200,26,0A,19,23 <CR>.**

14 The contents of byte locations 200, 201, 202 and 203 are changed to 26, 0A, 19 and 23, respectively.

F (FILL)

Fills an area of memory with specified byte values.

Syntax

F range,bytevalue[,bytevalue...]

range

the range of addresses whose contents are to be overwritten with the specified bytevalues. If only the offset is specified, then the segment indicated by the DS register is assumed.

bytevalue

a two digit hexadecimal value that is to overwrite the contents of the specified address(es).

Comments

- If the specified range contains more bytes than the list of byte values, then the list of byte values is repeated until the specified range is filled.
- If the list of byte values is longer than the specified range, the extra byte values are ignored.

Example

1 Enter **F04BA:100L 100,42,45,48,37,20<CR>.**

2 DEBUG fills memory locations 04BA:100 to 04BA:1FF with the byte values specified. The five values are repeated until all 256 locations are filled.

G (GO)

Executes the program currently in memory, optionally halting at specified breakpoint(s) and displaying information about the system and program environment.

Syntax

G [=address][,address...]

=address

the address in memory at which program execution is to start. "=" must be entered to distinguish a start address from a breakpoint address.

address

the breakpoint address. You can specify up to ten breakpoints, in any order.

Comments

If you enter G without parameters, the program currently in memory is executed starting from the address specified by the CS and IP registers.

If you specify the =address parameter, the contents of the CS and IP registers are changed to those specified by the =address parameter and the program in memory is executed, starting from the address you specified.

If you specify one or more breakpoint addresses, program execution stops at the first such address encountered and displays the contents of the registers, the state of the flags and the next instruction to be executed (see the Register command for a description of the display).

- If only an offset is entered for an address, the GO command assumes the segment in the CS register.

- If you enter more than ten breakpoints, DEBUG will display

  BP Error

- Before executing the program, the GO command replaces the contents of the breakpoint locations with an interrupt instruction (hexadecimal CC). Therefore, each breakpoint address that you specify must point to the first byte of an 8086 instruction, or unpredictable results occur.

  When program execution halts at a breakpoint DEBUG restores the original values of all the specified breakpoint locations. However, if the program terminates normally (that is, not at a specified breakpoint), the original values are not restored.

  Note: Once a program has reached completion (DEBUG has displayed "Program terminated normally") you must reload the program before you can re-execute it.

  The stack segment must have six bytes available at the stack pointer for this command, otherwise unpredictable results occur. This is because the GO command jumps into the user program with the IRET instruction. The flag, CS, and IP registers have to be pushed onto the stack in preparation for the IRET, taking up six bytes.

Example

1 Enter **G=200,1AF,141 <CR>.**

2 The program currently in memory is executed starting from location 200. Assuming location 141 is encountered before 1AF, then the program halts at location 141 and the register and flag values are displayed along with the next instruction to be executed. If neither breakpoint location is encountered, then the program terminates normally.

3 Enter **G<CR>.**

4 If, in step two, the program had halted at location 141, then program execution continues from that address.

H (HEXARITHMETIC)

Calculates and displays the sum and the difference of two hexadecimal values.

Syntax

H value_a,value_b

value_a

The first of two hexadecimal values.

value_b

The hexadecimal value that is to be added to or subtracted from value_a.

Comments

The hexadecimal values may be up to four digits long.

The Hex command displays two four-digit values:

— the first is the result of adding value_b to value_a

— the second is the result of subtracting value_b from value_a

Example

1. Enter **H19F,10A<CR>**
2. DEBUG displays

   02A9 0095
3. Enter **HFFFF,2<CR>.**
4. DEBUG displays

   0001 FFFD

I (INPUT)

Inputs and displays (in hexadecimal) one byte from the specified port.

Syntax **I value**

value the address of the port that the byte is to be input from.

Comments The port address can be up to 16 bits.

Example

1. Enter **I2F8.**
2. the byte at the addressed port is input and displayed.

L (LOAD)

Loads a file or absolute disk blocks into memory.

Syntax **L [address[,drive,block,count]]**

address the address in memory at which the file or range of blocks is to be loaded. If only an offset is entered, then the segment indicated by the CS register is assumed.

drive the drive from which disk blocks are to be loaded. For drive A you must enter 0, for drive B you must enter 1, etc.

block the first of a range of blocks to be loaded from the disk specified by the drive parameter.

count the number of blocks to be loaded.

Comments

- If all parameters are specified, then DEBUG loads blocks of information from disk into memory.
- If you enter L without parameters, or with just the address parameter, the file whose file control block is correctly formatted at location CS:5C is loaded into memory. The file control block at CS:5C is set either to the filespec specified when the DEBUG command was invoked, or to the filespec specified by the most recent "Name" command.
- The default location for programs to load is at CS:100. If you specify L and the address parameter, the file is loaded at the specified address unless it is a .EXE or .HEX file. In any case DEBUG sets the BX:CX registers to the number of bytes loaded.

- If the file has an .EXE extension, then it is relocated to the load address specified in the header of the .EXÈ file. That is, the address parameter to the Load command is ignored. The header itself is stripped off the .EXE file before the file is loaded into memory. Thus the size of the .EXE file on disk will differ from its size in memory.

- If the file is a .HEX file, entering the Load command with no parameters causes the file to be loaded starting at the address specified within the .HEX file. If the address parameter, however, is specified, then loading starts at the address which is the sum of the address specified and the address in the .HEX file.

Examples

The following examples assume the system to be initially in MS-DOS.

1 Enter **debug <CR>**
Nb:file.com <CR>
L <CR>.

Debug is entered and the subsequent Name command sets the file control block at CS:5C to identify file "file.com" on the diskette inserted in drive B. The Load command then loads this file into memory starting at CS:100 (the default address).

2 Enter **debug b:file.com** <**CR**>
L300 <**CR**>.

file.com is loaded into memory at location CS:100 by the DEBUG command. It is then relocated to CS:300 by the Load command.

M (MOVE)

Moves the contents of a specified range of memory addresses to the locations starting at a specified address.

Syntax

M range,address

range

The area of memory whose contents are to be moved. If you only enter an offset, the segment indicated in the DS register is assumed.

address

The start of the destination area. If you only enter an offset, then the segment indicated by the DS register is assumed.

Comments

If the source and destination areas overlap, the move is performed without loss of data.

The contents of the source area are not changed by the move, unless the destination area overlaps it.

If you specify an address as the end of the range, you must only enter the offset. The segment specified, or defaulted to, in the start address of the range is assumed.

Example

1 Enter **MCS:100,110,CS:500 <CR>** or **MCS:100L11,CS:500 <CR>.**

2 The 17 bytes starting at location CS:100 are copied to the 17 bytes starting at location CS:500.

N (NAME)

Provides filenames for the Load and Write commands or filename parameters for the program to be debugged.

Syntax

N filename(,filename...)

filename

the filename of a file to be loaded into memory, written to diskette, or used as a parameter to the file currently in memory.

Comments

The Name command can be used to provide:

the name of the disk file to be loaded into memory by a subsequent Load command.

the name to be assigned to the file currently in memory when the file is subsequently written to disk.

filename parameters to the file in memory to be debugged.

The first case enables you to specify the file you wish to debug after entering the DEBUG environment. That is, you can enter DEBUG without specifying parameters, then use the Name command to name the disk file you wish to debug, then load the file into memory using the Load command. This has the same effect as entering the filename as the first parameter to the DEBUG command upon invocation. In either case the file control block for the file to be debugged is set up at location CS:5C and the file is loaded.

In the second case, the file is already in memory and the Name command sets up the file control block for the specified filename at location CS:5C. When a Write command is subsequently entered the file in memory is written to disk with the filename whose file control block is set up at location CS:5C.

In the third case, the Name command provides filename parameters for the program currently in memory. Whatever file control block was set at CS:5C is replaced by that of the first such parameter. If a second file parameter is specified, its file control block is set up at location CS:6C. Only two file control blocks are set up, although additional filename parameters may be included if required. All the specified — including any delimiters and switches that may have been typed — are placed in a save area at CS:81, with CS:80 containing a character count. Parameters specified in this way are analogous to filenames specified in the argument list to the DEBUG command.

Examples

1 Enter **DEBUG <CR>**
Nb:file.com <CR>
L <CR>.

The system enters the DEBUG environment and FILE.COM resident on drive B has its file control block set up at location CS:5C. The Load command subsequently loads this file into memory.

This sequence has the same effect as entering

"DEBUG b:file.com"

2 Enter **Nb:newfile.com<CR>**
W<CR>

The file-control block is set up at location CS:5C for the file specifier "b:newfile.com". The subsequent Write command writes the file currently in memory to drive B and names the file "newfile.com".

3 Enter **DEBUG b:file 1.com <CR>**
Nfile2.dat,file3.dat<CR>
G<CR>

The DEBUG command loads the file named "file1.com" from drive B to be debugged. The Name command sets up two file control blocks at locations CS:5C and CS:6C for the file specifiers b:file2.dat and b:file3.dat, respectively. These files then become parameters to file1. COM when the subsequent GO command executes file1.COM. Therefore, the file is executed as if the following command line had been typed:

b:file 1 file2.dat file3.dat

O (OUTPUT)

Sends a specified byte to an output port.

Syntax **O value,byte**

value the address of the output port. It must be specified in hexadecimal and can be up to 16 bits.

byte a two-digit hexadecimal value to be sent to the specified port.

Example

1 O1E, 27 <CR>

2 the byte value 27H is output to the port 1EH.

Q (QUIT)

Terminates the DEBUG program.

Syntax **q**

Comments The Quit command terminates the debugger without saving the file you are working on. Control is returned to MS-DOS command mode.

R (REGISTER)

Displays the contents of the registers and flag settings, or displays the contents of a specified register with the option to change that value, or displays the flag settings with the option of reversing any number of those settings.

Syntax

R [register-name pipe: F]

register-name

any valid register name whose contents are to be examined and optionally changed. This may be one of:

AX	DX	SI	ES	IP
BX	SP	DI	SS	PC
CX	BP	DS	CS	

Note: IP and PC both refer to the Instruction Pointer.

F

the flag settings are to be displayed and optionally changed.

Comments

If you enter R without parameters, then the contents of all registers are displayed along with the flag settings and the next instruction to be executed. For example:

```
AX=058D BX=0000 CS=0000 DX=0000
SP=FFFO BP=0000 SI=0000 DI=0000 DS=058D
ES=058D CS=058D IP=013B
NV UP EI PL NZ NA PO NC
058D:013B 83D8      MOV DS,AX
```

If you enter R with a register name, then DEBUG displays the contents of that register. The command then waits for you to do one of the following:

— press <CR> to terminate the Register command without changing the value of the displayed register.

— change the value of the register by entering the four-digit hexadecimal value, then terminate the Register command by entering <CR>.

The valid flag values are shown in the following table:

Flag Name	Set	Clear
Overflow	OV (yes)	NV (no)
Direction	DN (decrement)	UP (increment)
Interrupt	EI (enabled)	DI (Disabled)
Sign	NG (negative)	PL (plus)
Zero	ZR (yes)	NZ (no)
Auxiliary Carry	AC (yes)	NA (no)
Parity	PE (even)	PO (odd)
Carry	CY (yes)	NC (no)

If you enter RF, then the current flag settings are displayed. You can then either

— press <CR> to terminate the Register command without changing the flag values, or

— change the setting of one or more flags by entering the alternate value of the appropriate flags. The new values may be entered in any order, with or without delimiters.

Example

1. Enter **R<CR>.**
2. DEBUG displays the contents of all registers, flag settings and the next instruction to be executed.
3. Enter **RIP <CR>.**
4. DEBUG displays the contents of the Instruction Pointer. For example:

   IP 0139

   :—
5. Enter **0138<CR>.**
6. The contents of the Instruction Pointer are changed to 0138.
7. Enter **RF <CR >.**
8. DEBUG displays the flag settings. For example:

   NV UP EI PL NZ NA PO NC.
9. Enter **PE ZR DI NG <CR>.**
10. The Parity flag is set to even (PE), the Zero flag is set (ZR), the Interrupt flag is cleared (DI), and the Sign flag is set (NG).
11. Enter **RF< CR>.**
12. DEBUG displays the new state of the flags

    NV UP DI NG ZR NA PE NC-

S (SEARCH)

Searches a specified range for a list of bytes.

Syntax

S range,list

range

the range of addresses within which the search is to be made. If you only enter the offset, the segment indicated by the DS register is assumed.

list

the list of one or more bytes to be searched for. Bytes in the list must be separated by a space or a comma.

Comments

For each occurrence of the list of bytes within the specified range, DEBUG returns the address of the first byte. If no address is returned, no match was found.

Example

1 Enter **/S100L100,20<CR>** or **S100,1FF,20<CR>.**

2 DEBUG displays the address of every occurrence of byte value 20 in the address range 100 to 1FF, inclusive, for example:

```
058D:010C
058D:0110
058D:0115
058D:0118
058D:0128
058D:01CE
```

T (TRACE)

Executes one or more instructions and displays the register contents, flag settings and the next instruction to be executed.

Syntax **T[=address][,value]**

=address DEBUG is to commence execution at this address.

value the number of instructions to be executed.

Comments If the =address parameter is not specified, execution begins at CS:IP.

If the value parameter is not specified, only one instruction is executed.

The display generated is of the same format as that of the Register command (without parameters).

Example

1. Enter **T = 200,5<CR>**.
2. Five instructions, starting with the one at location CS:200, are executed, and the register and flag values following each instruction are displayed along with the next instruction to be executed.
3. Enter **T<CR>**.
4. The instruction pointed to by CS:IP is executed and the register and flag contents are displayed along with the next instruction to be executed.

U (UNASSEMBLE)

Disassembles strings of bytes in memory and displays them as assembler-like statements along with their corresponding addresses.

Syntax

U [range]
or
U [address]

range the range of addresses whose byte values are to be disassembled. If you do not specify the segment, then the segment indicated by the CS register is assumed.

address the start of a 32 byte area of memory to be disassembled. If you only enter an offset, then the segment indicated by the CS register is assumed.

Comments

- If neither the range nor address parameter is specified, then 32 bytes are disassembled starting at location CS:IP. If the Unassemble command is given more than once, each subsequent invocation starts at the address following the last disassembled location.
- The number of bytes disassembled may be slightly more than the number you specified. This is because instructions are not always the same length and the final address in a range will not always contain the last byte of an instruction.
- The first address of a range, or the address parameter, must always refer to the first byte of an 8086 instruction, otherwise results are unpredictable.

Example

1 Enter **U058D:204L8 <CR>.**

2 Eight bytes starting at location 058D:204 are disassembled and the result displayed:

```
058D:0204   8D16DFOD   LEA   DX,[ODDF]
058D:0208   42         INC   DX
058D:0209   03DO       ADD   DX,AX
058D:020B   8916E50B   MOV   [OBE5],DX
```

W (WRITE)

Writes the file being debugged to disk.

Syntax **W [address[,drive,block,count]**

address the start address of the code in memory that is to be written to disk. If you enter only an offset, then the segment indicated in the CS register is assumed.

drive the drive containing the specified blocks to which code in memory is to be written. For drive A you must enter 0, for drive B you must enter 1, etc.

block the block number on disk that is the first of a contiguous range of blocks to be overwritten with code from memory.

count the number of disk blocks to be overwritten with code from memory.

Comments

- If you enter the WRITE command without parameters, then the file is written to disk starting from memory address CS:100. If you specify the address parameter, then the file in memory, starting from the specified address, is written to disk.

- In either case, before executing the WRITE command, BX:CX must be set to the number of bytes to be written if the count parameter is not included. This value was set up correctly when the file was loaded (either by the Load command or the DEBUG command itself). However, if, since loading the file, you have executed a GO or TRACE command, then the value of BX:CX will have been changed. Be sure this value is set up correctly.

- When the WRITE command writes a file to disk, it obtains the drive specifier and filename via the file control block set up at CS:5C. If no drive specifier is set up, then the default is assumed. This file control block is set up either by the DEBUG command (for the file you specify as a parameter to DEBUG) or by a subsequent NAME command. If it does not indicate the file specifier you require, you must set up this file control block using the NAME command. Refer to "Memory Maps, Control Blocks, and Diskette Allocation" in the System Programmers Manual for further details.

- When the file is written to disk it overwrites the version currently on disk unless the specified file name does not exist, in which case a new file is created.

- If all parameters are specified, then the code in memory is written to the drive specified by the parameter. The data to be written starts at the memory location specified by the address parameter, and is written to the blocks on the disk specified by the block and count parameters. Be extremely careful to correctly specify the blocks, since information stored there previously will be destroyed by this operation.

Examples

1 Enter **W <CR>**.

The file in memory, starting from location CS:100, is written to disk with the file specifier defined by the file control block set up at location CS:5C. The number of bytes written is given by BX:CX.

2 Enter **W200 <CR>**.

The file in memory, starting from location CS:200, is written to disk with the file specifier defined by the file control block set up at location CS:5C. The number of bytes written is given by BX:CX.

3 Enter **W200,1,1F,20 <CR>**.

Blocks 1F through 3F on drive B are overwritten with the data starting at memory location CS:200.

DEBUG ERROR MESSAGES

BF Bad Flag
You attempted to alter a flag, but entered some characters that are not acceptable pairs of flag values. See R (Register) command for the list of acceptable flag entries.

BP Too many Breakpoints
You specified more than ten breakpoints as parameters to the GO command. Reenter the command with ten or fewer breakpoints.

BR Bad Register
You entered the R command with an invalid register name.

DF Double Flag
You entered two values for one flag.

E CONFIG.SYS

- Overview
- Config.Sys Commands
  - Break
  - Buffers
  - Device
  - Files
  - Shell

Overview

CONFIG.SYS is a special file that MS-DOS uses to modify system parameters. Every time you start your system, MS-DOS searches for CONFIG.SYS in the ROOT directory on the default drive. If MS-DOS finds CONFIG.SYS, it executes the commands it finds there.

Some applications programs require you to make modifications to CONFIG.SYS. If you have purchased special hardware, a "device driver" may need to be identified in CONFIG.SYS.

CONFIG.SYS Commands

The special file CONFIG.SYS is processed automatically when MS-DOS starts. As with the batch file AUTOEXEC.BAT, this processing is automatic. MS-DOS will simply look at the root directory to see if the file is there.

CONFIG.SYS is an ASCII text file that can be edited by EDLIN or any other text editor that produces ASCII files. Five commands can be used in the CONFIG.SYS file. Each command changes a system parameter.

- BREAK <ON |OFF>
  Changes the way MS-DOS checks for a CTRL/BREAK

- BUFFERS = xx
  Sets the number of data buffers that MS-DOS uses.

- DEVICE = [d:][path] filename
  Adds a nonstandard device driver to MS-DOS.

- FILES = xx
  Sets the number of files that can be simultaneously open.

- SHELL = [d:][path] <filename> [d:] <path> /P
  Specifies an alternate command processor.

Break

Normally, MS-DOS checks for a CTRL/BREAK only when it is doing input or output. Some programs do very little (if any) input or output for long periods of time. The BREAK ON command sets MS-DOS to check for a CTRL/BREAK when any MS-DOS function is called.

BREAK OFF resets the default so that MS-DOS only checks for a CTRL/BREAK during input and output. This command can be used to override a BREAK ON that was set by CONFIG.SYS.

Buffers

The number of buffers has an effect on both the speed of disk I/O and available memory. A larger number of buffers allocates more memory to the system for operations. This is highly desirable for systems with large amounts of RAM that will be used for database applications. It is highly undesirable for systems with minimal RAM that do little work with disk files.

The system default is BUFFERS = 10, which is adequate for most purposes and requires 5K of RAM. If your system will be doing a significant amount of data handling, especially on a hard disk, increasing BUFFERS would improve access times at a cost of only 512 bytes of internal memory for each additional buffer.

Device

When MS-DOS is first started, it loads all of the standard device drivers for the keyboard, screen and so on. If your system requires a special device driver, this command tells MS-DOS where to find it. The device driver will then be loaded as an extension to MS-DOS.

Device drivers are .COM files with a specific structure described in Chapter 9 in the System Programmer's Guide. The command DEVICE = ANSI.SYS causes MS-DOS to replace the standard display and keyboard device drivers with the extended screen and keyboard support that the extended functions require.

If you wish to load several special device drivers, you must use a DEVICE command for each one.

Files

Opening a file with an ASCIIZ string eliminates the traditional direct handling of a File Control Block (FCB). MS-DOS will handle all these things for you by creating and maintaining FCBs internally. To do this, it needs memory. Each file requires 39 bytes of memory.

The FILES command sets aside memory for this operation. The default is FILES = 12. The maximum is FILES = 99. This is the limit for the entire system. A particular process (or program) can still have only 20 files open at once.

Shell

The COMMAND.COM file that MS-DOS uses as its "front end" processor can be replaced by another command processor. The SHELL command specifies the file to be used and the default path for processing commands. The command processor must be able to read and execute commands, and handle interrupts 22H, 23H, and 24H.

Since COMMAND.COM handles internal commands, .BAT file execution, and .EXE file loading, these functions will be unavailable unless the new command processor duplicates them.

F

EXE2BIN

- Syntax
- Example
- Discussion
- Messages

Syntax

EXE2BIN <input filename> <output file name>

Both file names are in the form:

[d:][path]filename

Example

EXE2BIN B:PROG.EXE B:PROG.COM

Discussion

In specifying the input file, everything except the file name is optional. If you do not specify a drive, the default is used. If you do not specify a path, the default path is used. If you do not specify an extension, the default is .EXE. The input file is converted to .COM file format (memory image of the program) and placed in the output file.

You are not required to enter any part of the output file specification. If you do not specify a drive, the drive of the input file will be used. If you do not specify an output path or filename, the input path or filename will be used. If you do not specify a filename extension in the output filename, the new file will be given an extension of .BIN.

The input file must be in valid .EXE format produced by the linker. The resident, or actual code and data part of the file must be less than 64K. There must be no STACK segment.

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

1. 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 address fixups (that is, instructions that contain a reference to an absolute segment address) are allowed, as .COM files must be segment relocatable. Once the conversion is complete, rename the resulting file with a .COM extension. The command processor can load and execute the program in the same way as the .COM programs supplied on your MS-DOS diskettes.

2. If CS:IP is not specified in the .EXE file, a pure binary conversion is assumed. If segment fixups are necessary (i.e., the program contains instructions requiring a segment address), you are prompted for the fixup value. This value is the absolute segment at which the program is to be loaded. The resulting program is usable only when loaded at the absolute memory address specified by your application. The command processor is not capable of properly loading the program. This is the case when writing a .BIN program to use in an application such as a device driver that is always loaded at the same absolute address.

EXE2BIN Messages

Amount read less than size in header

The program portion of the file was smaller than indicated in the file's header. You should reassemble and relink your program.

File cannot be converted

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

File creation error

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

File not found

The file is not on the diskette specified.

Fixups needed — base segment (hex):

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

Insufficient disk space

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

Insufficient memory

There is not enough memory to run EXE2BIN.

WARNING — Read error in EXE file

Amount read less than size in header. This is a warning message only. However, it is usually a good idea to reassemble and relink your source program when this message appears.

G ANSI.SYS

An ANSI Escape Sequence is a series of characters, beginning with an ESCAPE character, that you can use to define functions to MS-DOS. You can reassign keys, change graphics functions, and affect cursor movement.

There are a variety of applications for the extended keyboard and screen support afforded by the ANSI Escape Sequences. Some of the most common uses are:

- You may be engaged in communications with a host computer that expects ANSI Escape Sequences (i.e., your PC 6300 is emulating a terminal).
- You may reassign a function key so that, for example, pressing F9 clears the screen.
- Some applications software that you purchase may make use of these sequences.

Installing the ANSI.SYS Device Driver

To utilize the ANSI Escape Sequences, load the special device driver, ANSI.SYS, by placing a command in the CONFIG.SYS file.

Put this statement in CONFIG.SYS:

DEVICE = ANSI.SYS

Every time you start MS-DOS, the ANSI.SYS driver will be loaded into memory so that you can use the escape sequences.

Note:
In the following syntax diagrams, replace the number sign (#) with the appropriate ASCII digits.

If you do not specify a value, or specify a value of zero, the default will be used.

In the control sequences below, ESC is the 1-byte code for ESC (hex 1B).

Cursor Position (CP)

Syntax **ESC[#;#H**

Moves the cursor to the position specified by the parameters. The first parameter specifies the line number and the second parameter specifies the column number. The default value is one. If no parameter is given, the cursor moves to the home position.

Cursor Up (CU)

Syntax **ESC[#A**

Moves the cursor up one line without changing columns. The value of # determines the number of lines moved. The default for # is one. This sequence is ignored if the cursor is already at the top line.

Cursor Down (CD)

Syntax **ESC[#B**

Moves the cursor down one line without changing columns. The value of # determines the number of lines moved. The default value for # is one. This sequence is ignored if the cursor is already at the bottom line.

Cursor Forward (CF)

Syntax **ESC[#C**

Moves the cursor forward one column without changing lines. The value of # determines the number of columns the cursor will move. The default value for # is one. This sequence is ignored if the cursor is already in the rightmost column.

Cursor Backward (CB)

Syntax **ESC[#D**

Moves the cursor back one column without changing lines. The value of # determines the number of columns the cursor will move. The default value of # is one. This sequence is ignored if the cursor is already in the leftmost column.

Horizontal and Vertical Position (HVP)

Syntax **ESC[#;#f**

Moves the cursor to the position specified by the parameters. The first parameter specifies the line number and the second parameter specifies the column number. The default is one. If no parameter is given, the cursor is moved to the home position.

Device Status Report (DSR)

Syntax **ESC[6n**

The console driver outputs a CPR (Cursor Position Report) sequence upon receipt of DSR. See below.

Cursor Position Report (CPR)

Syntax **ESC[#;#R**

The CPR sequence reports the current cursor position through the standard input device. The first parameter specifies the current line; the second parameter specifies the current column.

Save Cursor Position (SCP)

Syntax **ESC[s**

The current cursor position is saved. This cursor position can be restored with the RCP (Restore Cursor Position) sequence.

Restore Cursor Position (RCP)

Syntax **ESC[u**

Restores the cursor to the value it had when the console driver received the SCP (Save Cursor Position) sequence.

Erase in Display (ED)

Syntax **ESC[2J**

Erases the entire screen. Cursor moves to the home position.

Erase in Line (EL)

Syntax **ESC[k**

Erases from the cursor to the end of the line and includes the cursor position.

Set Graphics Rendition (SGR)

Syntax

ESC [#;...;#m

Sets the character attribute specified by the parameter(s). All characters that follow will have the attribute specified until the next occurrence of this sequence.

Parameter	Meaning
0	All attributes Off (normal white on black)
1	Bold On (high intensity)
4	Underscore On (monochrome display only)
5	Blink On
7	Reverse Video On
8	Cancelled On (invisible)
30	Black foreground
31	Red foreground
32	Green foreground
33	Yellow foreground
34	Blue foreground
35	Magenta foreground
36	Cyan foreground
37	White foreground
40	Black background
41	Red background
42	Green background
43	Yellow background
44	Blue background
45	Magenta background
46	Cyan background
47	White background

Set Mode (SM)

Syntax 1 **ESC[=#h**

Syntax 2 **ESC[=h**

Syntax 3 **ESC[=0h**

Syntax 4 **ESC[?7h**

Invokes the screen width or type corresponding to the parameter.

Parameter	Meaning
0	40 × 25 black and white
1	40 × 25 color
2	80 × 25 black and white
3	80 × 25 color
4	320 × 200 color
5	320 × 200 black and white
6	640 × 200 black and white
7	wrap at end of line

Reset Mode (RM)

Syntax 1 **ESC[=#I**

Syntax 2 **ESC[=I**

Syntax 3 **ESC[=0I**

Syntax 4 **ESC[?7I**

Parameters are the same as for Set Mode (SM) except that parameter 7 resets wrap to end-of-line mode (characters past the end of line are thrown away).

Syntax 1 **ESC[#;#;...#p**

Syntax 2 **ESC["string";p**

Syntax 3 **ESC[#;"string";#;#;"string";#p**

The first ASCII code in the control sequence defines which key is being mapped. The remaining numbers define the sequence of ASCII codes generated when this key is intercepted. However, if the first code in the sequence is zero (NUL), then the first and second codes make up an extended ASCII redefinition. (See the chapter on "ROM BIOS Service Routines" in the "System Programmer's Guide" for more information.)

Examples

- Reassign the F9 key to do a CLS command followed by a carriage return.

  ESC[0;67;"CLS";13p

- Map a lowercase e to upper case.

  ESC[101;69p

H RAMDISK

- Overview

Overview

RAMDISK is a device driver that lets you use memory as if it were disk. This improves your system's performance.

You can see an especially big improvement in performance by loading assemblers and compilers into RAMDISK.

The RAMDISK.DEV file is a device driver that simulates a disk drive by using a portion of your system's memory.

The RAMDISK operates at the system's memory speed.

You can install more than one RAMDISK. The drive letters are automatically assigned. For example, if your system has one floppy diskette drive and one hard disk drive, the floppy drive is referred to as A and the hard disk as C. The first RAMDISK is referred to as D, the second as E, and so on.

You can specify the amount of system memory that is to be used by the RAMDISK. The default size is 8K.

Note:

> The contents of the RAMDISK are lost when you restart the system or turn off the system's power. Use the COPY command to move files from RAMDISK to an actual disk if there are files you want to save.

Installing the RAMDISK

To install the RAMDISK device driver, include the following statement in the CONFIG.SYS file:

device = [d:][pathname]ramdisk.dev [nnn]

nnn is the size of virtual RAMDISK in K bytes (1024). The default value is 8K bytes.

INDEX

D

E

F

G

H

I

Q

R

S

T

U

V

W

LIMITED USE SOFTWARE LICENSE AGREEMENT

THIS CARD CONTAINS THE AT&T INFORMATION SYSTEMS, INC. ("AT&T-IS") LIMITED USE SOFTWARE LICENSE AGREEMENT.

YOU SHOULD READ THE TERMS AND CONDITIONS OF THIS LICENSE BEFORE YOU OPEN THE SEALS ON THE PACKAGES CONTAINING THE DISKETTE AND THE DOCUMENTATION. ONCE YOU HAVE READ THIS LICENSE AGREEMENT AND AGREE TO ITS TERMS, YOU MAY OPEN THE SEALED ENVELOPE CONTAINING THE PROGRAM MEDIA. BY OPENING THE SEAL, YOU SHOW YOUR ACCEPTANCE OF THE TERMS OF THIS LIMITED USE SOFTWARE LICENSE AGREEMENT. THE AGREEMENT IS IN EFFECT FROM THEN UNTIL YOU RETURN ALL THE SOFTWARE TO AT&T-IS.

IN THE EVENT THAT YOU DISAGREE WITH ANY OF THE TERMS OF THIS AGREEMENT, RETURN YOUR SALES RECEIPT AND THE SOFTWARE WITH THE SEAL UNBROKEN TO THE LOCATION WHERE YOU OBTAINED THE SOFTWARE AND YOUR MONEY WILL BE REFUNDED.

LIMITED USE SOFTWARE LICENSE AGREEMENT

AT&T Information Systems, Inc. ("AT&T-IS") and you agree that the terms and conditions hereof will apply to the SOFTWARE supplied herewith and derivatives obtained therefrom, including any copy. The term SOFTWARE includes programs and related documentation supplied herewith.

If you have executed a separate Software Agreement with AT&T-IS covering the Software supplied herewith, such Software Agreement will govern.

1. TITLE AND LICENSE GRANT

The SOFTWARE is copyrighted and/or contains proprietary information protected by law. All SOFTWARE will remain the sole property of AT&T-IS or its suppliers. AT&T-IS hereby grants you a personal, non-transferable and non-exclusive right to use, in the United States, Puerto Rico and Canada, all SOFTWARE, in whatever form recorded, which is furnished to you under or in contemplation of this Agreement. This grant is limited to use on the single host processor that you indentify on the enclosed Software Registration Card. Use of this SOFTWARE by you on any processor other than the one identified by you or removal of the SOFTWARE from a country in which use is licensed shall automatically terminate this license.

You agree to obtain prior AT&T-IS approval for multi-processor usage.

You agree to use your best efforts to see that any user of the SOFTWARE licensed hereunder complies with the terms and conditions of this Agreement and refrains from taking any steps, such as reverse assembly or reverse compilation, to derive a source code equivalent of the software.

2. SOFTWARE USE

A. You are permitted to make a single archive copy, provided the SOFTWARE shall not be otherwise reproduced, copied, or, except for the documentation, disclosed to others in whole or in part, and provided that the archive copy shall contain the same copyright notice and proprietary marking including diskette markings; as appears on the original SOFTWARE.

B. The SOFTWARE,

1. together with any archive copy thereof, shall be either returned to AT&T-IS or destroyed when no longer used with the host processor for which it was initially furnished, or when the license to use is terminated; and
2. shall not be removed from a country in which use is licensed.

3. LIMITED WARRANTY

A. AT&T-IS warrants that the SOFTWARE will be in good working order and will replace, without charge, any SOFTWARE which is not in good working order if returned to the location where you obtained it within (90) days of delivery to you. At its option, AT&T-IS may refund the purchase price of the SOFTWARE.

B. AT&T-IS does *not* warrant that the functions of the SOFTWARE will meet your requirements or that SOFTWARE operation will be error-free or uninterrupted.

C. AT&T-IS has used reasonable efforts to minimize defects or errors in the SOFTWARE. HOWEVER, YOU ASSUME THE RISK OF ANY AND ALL DAMAGE OR LOSS FROM USE, OR INABILITY TO USE THE SOFTWARE.

D. Unless a separate agreement for software maintenance is entered into between you and AT&T-IS, AT&T-IS bears no responsibility for supplying assistance for fixing or for communicating known errors to you pertaining to the SOFTWARE supplied hereunder.

E. YOU UNDERSTAND THAT, EXCEPT FOR THE 90 DAY LIMITED WARRANTY RECITED ABOVE AT&T-IS, ITS AFFILIATES, CONTRACTORS, SUPPLIERS AND AGENTS MAKE NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIM ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states or other jurisdictions do not allow the exclusion of implied warranties or limitations on how long an implied warranty lasts, so the above limitations may not apply to you. This warranty gives you specific legal rights and you may also have other rights which vary from one state or jurisdiction to another.

4. EXCLUSIVE REMEDIES AND LIMITATION OF LIABILITIES

A. YOU AGREE THAT YOUR SOLE REMEDY AGAINST AT&T-IS, ITS AFFILIATES, CONTRACTORS, SUPPLIERS, AND AGENTS FOR LOSS OR DAMAGE CAUSED BY ANY DEFECT OR FAILURE IN THE SOFTWARE REGARDLESS OF THE FORM OF ACTION, WHETHER IN CONTRACT, TORT, INCLUDING NEGLIGENCE, STRICT LIABILITY OR OTHERWISE, SHALL BE THE REPLACEMENT OF AT&T-IS FURNISHED SOFTWARE, PROVIDED SUCH SOFTWARE IS RETURNED TO AT&T-IS WITH A COPY OF YOUR SALES RECEIPT. THIS SHALL BE EXCLUSIVE OF ALL OTHER REMEDIES AGAINST AT&T-IS, ITS AFFILIATES, CONTRACTORS, SUPPLIERS OR AGENTS, EXCEPT FOR YOUR RIGHT TO CLAIM DAMAGES FOR BODILY INJURY TO ANY PERSON.

B. Regardless of any other provisions of this Agreement, neither AT&T-IS nor its affiliates, contractors, suppliers or agents shall be liable for any indirect, incidental, or consequential damages (including lost profits) sustained or incurred in connection with the use, operation, or inability to use the SOFTWARE or for damages due to causes beyond the reasonable control of AT&T-IS, its affiliates, contractors, suppliers and agents attributable to any service, products or action of any other person.

C. For SOFTWARE licensed in the United States and Puerto Rico, this Agreement shall be construed in accordance with and governed by the laws of the State of New Jersey. For SOFTWARE Licensed in Canada, this agreement shall be construed in accordance with and governed by the laws of Ontario.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT AND UNDERSTAND IT, AND THAT BY OPENING THE SEAL CONTAINING THE PROGRAM MEDIA YOU AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT THIS AGREEMENT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE RIGHTS AND LIABILITIES OF THE PARTIES. THIS AGREEMENT SUPERSEDES ALL PRIOR AGREEMENTS, PROPOSALS OR UNDERSTANDINGS, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

AT&T SOFTWARE SERVICE INFORMATION

All of the documentation for this AT&T Software Product has been carefully written to cover most all of the questions that may arise during normal use.

For assistance in areas not covered in the documentation, and requests for media replacement under the limited warranty, a toll free hotline is available during normal business hours. The hotline will assist with usage and problem diagnosis for a period of 90 days after the date of purchase, delivery, or AT&T's provided installation, whichever is latest **provided that a completed Software Registration Card is returned within 5 days** of that date. Returning the Software Registration card to AT&T will enable us to provide you with updates, as necessary.

For assistance call AT&T at: **800-922-0354.**

Or write:
AT&T Customer Systems Support Center
Department 5
P.O. Box 8355
Iselin, New Jersey 08830

When you call or write, be sure to provide the 16 character serial number of the AT&T Software Product. The serial number can be found on the media (disk, tape, or cartridge) label. Also mention:

- the date of purchase, delivery, or date of AT&T installation
- your name and telephone number
- computer location
- your company name
- how purchased (either directly from AT&T or through an authorized AT&T dealer)

Additional assistance programs are available from AT&T. These include hotline service and updates beyond the initial 90 day period as well as a variety of on-site options. Contact your AT&T Account Executive or call 800-922-0354 for further information.