



MICROSOFT®

**MS-DOS<sup>TM</sup>**  
**USER'S**  
**GUIDE**  
**WITH SUPPLEMENT FOR THE PCW-1**

version 2.11





MICROSOFT®

**MS-DOS™**

---

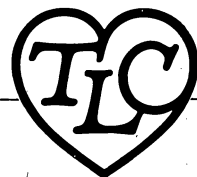
**USER'S**

---

**GUIDE**

**WITH SUPPLEMENT FOR THE PCW-1**

version 2.11



Information in this manual is subject to change without notice.

Copyright MINOLTA CAMERA CO., LTD. 1985. All rights reserved.

Copyright Microsoft Corporation, 1982, 1983.

IBM is a registered trademark of International Business Machines Corporation.

Microsoft is a registered trademark of Microsoft Corporation.

MS is a trademark of Microsoft Corporation.

Lotus 1-2-3 is a trademark of Lotus Development Corporation.

PCW-1 MS-DOS Version 2.11 Package Contents

\* 1 disk with the following files:

COMMAND	COM
FORMAT	COM
CHKDSK	COM
MORE	COM
RECOVER	COM
PRINT	COM
SYS	COM
DISKCOPY	COM
DEBUG	COM
SORT	EXE
FC	EXE
FIND	EXE
EXE2BIN	EXE
TPM	COM
EDLIN	COM

\* 1 manual

System Requirements

The MS-DOS for the PCW-1 disk is designed specifically for the Minolta Office System PCW-1.



# CONTENTS

## Section 1 Supplement for the PCW-1

### Chapter 1 Introduction

About this Guide . . . . .	1-2
What is "MS-DOS"? . . . . .	1-3
PCW-1 MS-DOS Commands . . . . .	1-4

### Chapter 2 PCW-1 Keyboard

Keyboard Labels . . . . .	2-2
Keyboard Functions	
Keys that Assume Precise Functions in PCW-1 MS-DOS	
Zero (0) and Oh (o) . . . . .	2-5
One (1) and El (1) . . . . .	2-5
Spacebar (Blank Spaces) . . . . .	2-5
Shift key . . . . .	2-6
Code key . . . . .	2-6
Return, Enter keys . . . . .	2-6
Backspace, Left Cursor keys . . . . .	2-6
Functions Requiring a Combination of Keys	
Break . . . . .	2-7
Pause . . . . .	2-7
Print Screen . . . . .	2-7
System Reset. . . . .	2-8

### Chapter 3 Files and Basic Operation

Basic Operation	
Starting PCW-1 MS-DOS	
Starting From the Word Processor . . . . .	3-2
Starting From Power OFF. . . . .	3-2
How to Get the DOS "Ready" Signal. . . . .	3-3
How to Change Disk Drives. . . . .	3-4
Exiting PCW-1 DOS . . . . .	3-5
Starting Application Software . . . . .	3-5
Files and Selected Commands	
File Types and Their Use. . . . .	3-7
How to Name MS-DOS Files. . . . .	3-7
How to Use MS-DOS Commands. . . . .	3-9
Selected MS-DOS Commands. . . . .	3-10
DIR (Directory). . . . .	3-11
DEL (Delete) . . . . .	3-13

COPY . . . . .	3-14
DISKCOPY . . . . .	3-15
REN (Rename) . . . . .	3-16
FORMAT . . . . .	3-17
TYPE . . . . .	3-18

**Section 2 Microsoft MS-DOS User's Guide (plus PCW-1 supplement)**

See Contents page in Section 2

**Section 3 Microsoft DEBUG (plus PCW-1 supplement)**

See Contents page in Section 3

**Section 1**

**Supplement for the PCW-1**

Chapter 1 Introduction

Chapter 2 PCW-1 Keyboard

Chapter 3 Files and Basic Operation

**Chapter 1**  
**Introduction**

About this Guide

What is "MS-DOS"?

PCW-1 MS-DOS Commands

### About This Guide

This guide is divided into three sections:

Section 1 - Supplement for the PCW-1

Section 2 - Microsoft MS-DOS User's Guide

Section 3 - Microsoft DEBUG Utility

Read Section 1 first. It supplies you with the information you need to work with PCW-1 MS-DOS.

Section 1 contains three chapters. Chapter 1 serves as an introduction, but it also provides areas that are crucial to both beginner and veteran computer users: the PCW-1 commands that are specific to PCW-1 MS-DOS. Chapter 2 illustrates keyboard labeling of PCW-1 which is very important for beginners and veterans. It then lists and explains the basic PCW-1 MS-DOS key commands. For those with limited computer experience, Chapter 3 describes basic PCW-1 MS-DOS operation that will allow any user to start the DOS system and to perform commands that will be used every day.

Those who have had some experience with MS-DOS can skip Chapter 3. Be sure, however, to pay close attention to "PCW-1 MS-DOS commands" and "Keyboard Labels" in Chapters 1 and 2. Beginning personal computer users should read this entire section carefully.

Section 2, Microsoft MS-DOS Guide, and Section 3, Microsoft DEBUG, cover all the functions of the PCW-1 MS-DOS. These sections are also tailored with PCW-1 MS-DOS information. All readers should keep in mind that some material in these sections expand upon material covered in Section 1.

The key labels in Sections 2 and 3 are those of personal computer keyboards. Please refer to the "Keyboard Labels" in this section when you need to know which key on the PCW-1 keyboard has that same function.

**What is "MS-DOS"?**

As you may have read in the **Look It Up** or **Getting Started** manuals that were packaged with your Minolta Office System PCW-1, "MS-DOS" stands for "Microsoft Disk Operating System." This disk contains sets of instructions, known as **programs**, for the computer to follow in order to be able to manage itself. This group of programs is called an **operating system**. Because these instructions are contained on a disk, as opposed to residing in the physical parts of the computer, it is referenced as a **disk operating system**.

The operating system is the liaison between the hardware parts of the computer (disk drives, screen, and printer, for example) and the software, such as your PCW-1 Word Processing System, Lotus 1-2-3, and other programs. MS-DOS is your "bridge" -- it allows you to start up the computer, create and delete files, and run and link programs.

**PCW-1 Word Processing and other proprietary software** require some of the operating system functions of MS-DOS. These functions are already contained on the PCW-1 disks and, therefore, you do not need to use the PCW-1 with its own software.

**Application software**, on the other hand, is a group of programs designed to do a specific job, like inventory, accounting, or database management. Lotus 1-2-3 is an example of application software, as well as computer games. Application software communicates with MS-DOS, which in turn communicates with the computer. You cannot run application software without the MS-DOS disk.

You will notice that we use the term "MS-DOS for the PCW-1", shortened to "PCW-1 MS-DOS" for convenience. This is to remind you of the fact that this particular MS-DOS disk is specially designed to work only with the Minolta Office System PCW-1.

**NOTE:** The PCW-1 is generally compatible with MS-DOS based programs that run on the IBM PC and other computers that conform to the same standard. Certain MS-DOS programs may require special graphics capabilities or more memory than the standard 256K available on the PCW-1. Before purchasing optional software packages, consult your authorized Minolta Dealer or Sales Representative for details.

**PCW-1 MS-DOS Commands**

The following commands are included in PCW-1 MS-DOS:

IO.SYS	hardware operating system interface
MS-DOS	operating system
COMMAND	command processor
CHKDSK	checks disks
DEBUG	debugger
DISKCOPY	backup utility
EDLIN	line editor
FIND	finds a string
FC	compares files
FORMAT	formats
EXE2BIN	converts EXE. files
MORE	reviews text
PRINT	print spooler
RECOVER	recovers disks
SORT	sorts text
TPM	tailors PCW-1's RS232C port

These commands operate in the same way on the PCW-1 as they do on other IBM PC and compatible machines with three exceptions. They are:

PRINT - - If you use the PT-1 Printer for the printout, see notes on page 5-33 of Section 2.

DEBUG - - See page 2-37 of Section 3.

TPM - - Functions in much the same way as IBM's MODE command. TPM customizes the RS232C communication port so that the PCW-1 can "speak" to another computer or computer system. For the necessary "how to" information, see the discussion of this command on page 5-64 of Section 2.

## Chapter 2

### PCW-1 Keyboard

#### Keyboard Labels

#### Keyboard Functions

##### Keys that Assume Precise Functions in PCW-1 MS-DOS

- Zero (0) and Oh (o)
- One (1) and El(1)
- Spacebar (Blank Spaces)
- Shift key
- Code key
- Return, Enter keys
- Backspace, Left Cursor keys

##### Functions Requiring a Combination of Keys

- Break
- Pause
- Print Screen
- System Reset



### Keyboard Labels

We know that the PCW-1's personal computing (PC) mode is designed to run MS-DOS compatible (also called "IBM compatible") application software; therefore, it should not surprise you to find that the PCW-1's keyboard closely resembles that of the IBM PC. Most of the differences between these systems are caused by PCW-1's triple capabilities of electronic typewriter, dedicated word processor, and personal computer.

Figures 2-1 and 2-2 on the following pages illustrate where the IBM PC "action" keys are located on the PCW-1 keyboard. Anyone familiar with IBM terminology who wants to use CAPS LOCK, for example, will know that he or she must hit CODE on the PCW-1. Please refer to these diagrams as you read about the PCW-1 keyboard.

### Function Keypad

The function keypad is controlled by each particular application program. The key labelled F1/Help functions as a HELP key with the Minolta Word Processing disk.

### Typing Keypad

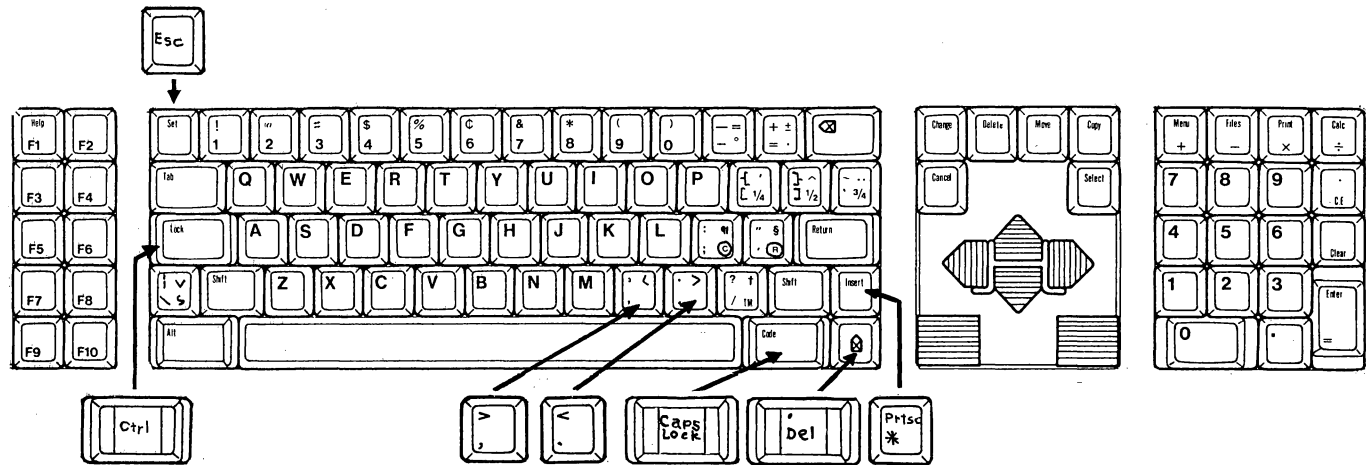
The PCW-1 keyboard assumes the keycap labels of the PC keyboard in personal computing mode. PCW-1 keys with more than two labels use only the left symbols in PC mode if not specifically shown on the figure 2-1.

### Editing Keypad

Generally not used in PC mode.

### Numeric Keypad

This keypad assumes the keycap labels of the PC keyboard in personal computing mode. In certain applications when cursor movement is active on this keypad, you may also use the cursor keys on the Editing Keypad. (See figure 2-3 on page 2-3.)



Function Keypad

Typing Keypad

Editing Keypad

Numeric Keypad  
 (See Figure 2-3 on the next page.)

Figure 2-1: PCW-1 Keyboard PT-1

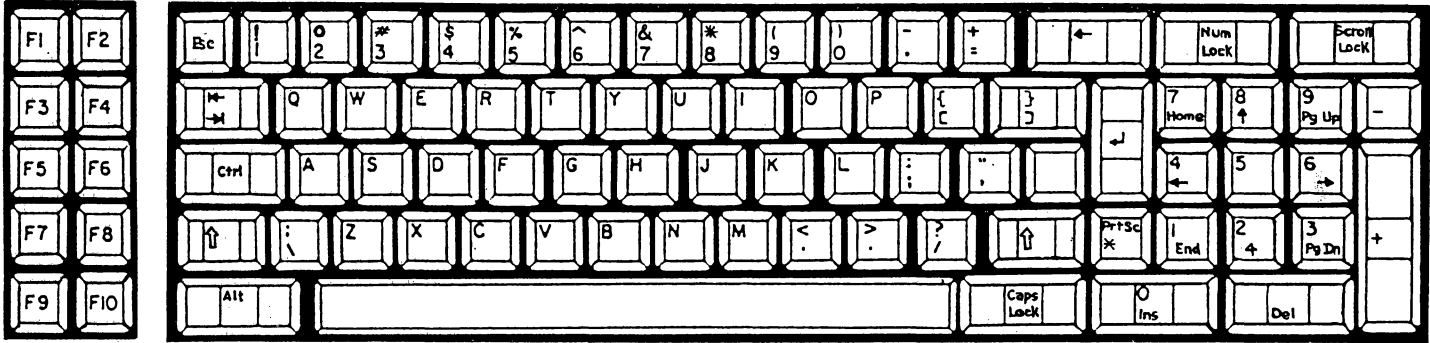


Figure 2-2: IBM PC Keyboard

Figure 2-3:  
 Numeric Keypad  
 functions in the  
 personal computer  
 mode

		Num Lock	SCROLL LOCK /BREAK
7 Home	8 ↑	9 Pg Up	-
4 ←	5	6 →	T
1 End	2 ↓	3 Pg Dn	
0 Ins		Del	↙

## Keyboard Function

This section is primarily for beginning users of personal computer keyboards. As you will see, the computer keyboard allows you to do many things that you cannot do with a typewriter.

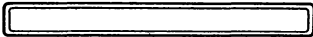
At times, the computer is less tolerant than your word processor. For example, hitting the wrong key can sometimes erase your work. Commonly used keys that operate in one way in PCW-1 MS-DOS and another way in word processing are described in the following sections.

### Keys that Assume Precise Functions in PCW-1 MS-DOS

**Zero ( 0 ) and Oh ( o )** These keys cannot be used interchangeably. The computer knows the difference and will not accept an o (oh) if a 0 (zero) is called for.

**One ( 1 ) and El ( l )** Do not use these keys interchangeably.

### Spacebar



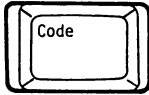
The spacebar is used to advance the cursor one space to the right, just as it is in typing. However, it is important to remember that when the computer's spacebar moves the cursor over an existing character, it eliminates that character and replaces it with a blank space.

Thus, you can use the PCW-1's spacebar to erase an entire line of text by holding the spacebar down moving the cursor from the left to the right.

**NOTE:**The computer reads a **blank space** as a character. This means that if you incorrectly insert or delete a blank in a command or a filename, you may receive an error message.

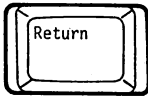


The **SHIFT** keys, located on the lower left and lower right sides of the typing keypad, are used to type capital (uppercase) letters and to type the upper position on all keys that contain two possible characters.

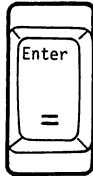



The **SHIFT LOCK** feature of typewriters is called **CAPS LOCK** in personal computer operations. For a situation that requires consistent use of uppercase letters, press the PCW-1 **CODE** key to toggle **CAPS LOCK** on and off.

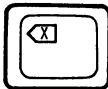
**CAPS LOCK** has a somewhat different effect than the Shift Lock of the PCW-1 Typewriter and Word Processor. With **CAPS LOCK** on, the typing keypad numbers still appear as numbers. To use the symbol keys with **CAPS LOCK on**, hold down the **SHIFT** key and type the desired symbol.




,



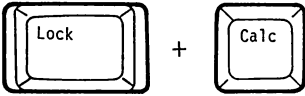
The PCW-1's **RETURN** ( on the IBM keyboard) and **ENTER** keys resemble the typewriter's return key, but they are much more powerful. The **RETURN** key and the **ENTER** key allow you to tell the computer that you have finished typing an entry. For example, if you are told to **enter** the word "the", then you must type "the" and press the **RETURN** or **ENTER** key. The computer will not react to the word until you **enter** it.



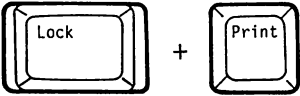
This key ( on the IBM keyboard) functions as a **character delete** key. At times you will type an entry and notice an error before you press the Return or Enter key. To correct this kind of simple typing error, use the **backspace** key, not the key 4 on the numeric keypad, to erase what you just typed.

Functions Requiring a Combination of Keys

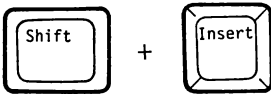
The following combinations of keys are used in performing personal computing operations:


**BREAK**

If you enter a command, but then need to stop the command from finishing, press and hold **LOCK** and **CALC** (Ctrl + Scroll Lock/Break of the IBM keyboard). Then release both keys. These two keys will "terminate" the command, which means that the command breaks off and returns you to the "ready" position, or "DOS prompt" (for more about the DOS prompt, see Starting and Exiting in Chapter 3).

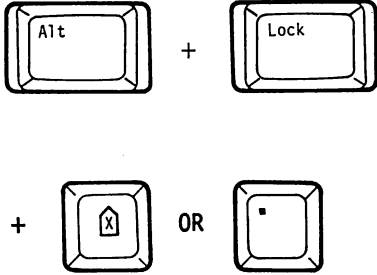
**PAUSE**



Press and hold the **LOCK** key, press the **PRINT** key (Ctrl + Num Lock on the IBM keyboard), and then release both keys to stop information from "scrolling" so quickly (moving up the screen until it disappears) that you can't read it. This is called the "pause" function. Strike any active key when you have finished reading and are ready for more information to scroll up the screen.

**PRINT SCREEN**

To print what is currently on the screen, such as the directory of a disk, insert paper into the printer and press the **SHIFT** and **INSERT** keys simultaneously (  + PrtSc keys on the IBM keyboard), and then release them to activate the printer.

## SYSTEM RESET



And perhaps most important, you may find yourself in a situation in which you would like to clear the screen and start PCW-1 MS-DOS again. This is called a "system reset" or "warm boot." You do this by making sure that your DOS disk is in Drive A; you then press and hold the ALT, LOCK, and  or  keys simultaneously (Ctrl + Alt + Del keys on the IBM keyboard). Then release all three. In a few moments you will see the DOS startup lines.

## Chapter 3

### Files and Basic Operation

#### Basic Operation

##### Starting PCW-1 MS-DOS

Starting From the Word Processor

Starting From Power OFF

How to Get the DOS "Ready" Signal

How to Change Disk Drives

##### Exiting PCW-1 DOS

##### Starting Application Software

#### Files and Selected Commands

File Types and Their Use

How to Name MS-DOS Files

How to Use MS-DOS Commands

Selected MS-DOS Commands

DIR (Directory)

DEL (Delete)

COPY

DISKCOPY

REN (Rename)

FORMAT

TYPE



This chapter is primarily for those who have not used MS-DOS keys or commands before. Instructions on starting PCW-1 MS-DOS will be followed by explanatory information on files and how to use selected commands.

More advanced information about files and commands are contained in Sections 2 and 3 of this manual.

## **Basic Operation**

### **Starting PCW-1 MS-DOS**

There are two ways to start the PCW-1 MS-DOS system: directly from the Word Processor, or with the machine power turned off.

#### **Starting From the Word Processor**

The process of starting the PCW-1 MS-DOS operating system is frequently referred to as "loading" or "booting". To load DOS from the Word Processor:

1. Press **MENU** to display the Main Menu.
2. Select **Run MS-DOS and Utilities** to display MS-DOS and Utilities Menu.
3. Select **Run MS-DOS**. The message "Insert a DOS disk into Drive A. Press ENTER to continue." is displayed at the bottom of the screen.
4. Insert your PCW-1 MS-DOS disk in Drive A, and press **ENTER**.
5. At this time, the DOS startup lines; asking you to supply the correct date and time, appear on the screen. Go to **How to Get the DOS "Ready" Signal** following the topic below.

#### **Starting From Power Off**

When you start DOS with the machine power turned off, you are "loading" or "cold booting" the system. To load DOS:

1. Insert the DOS disk into Drive A and turn the power switch on.
2. After a few moments the screen will display some information about the system.
3. It will (as above) ask you to provide the date and time. Go to **How to Get the DOS "Ready" Signal** next.

## How to Get the DOS "Ready" Signal

Before you can perform any operations with DOS, you have to arrive at the DOS "ready" signal. More commonly referred to as the "DOS prompt" or the "A prompt", this signal indicates that DOS is ready to use. It also indicates that you will begin your operations using the disk in Drive A. The following characters represent the DOS prompt:

A>\_

To get this prompt after loading DOS, you have the option of either bypassing the startup lines (date and time) or by entering the date and time.

### Entering date and time

1. The first startup line asks you to "Enter new date:\_"
2. To enter the date, type the date in this format:

month/day/year  
or  
month-day-year

3. Press RETURN.
4. You are now asked to "Enter new time:\_"
5. To enter the time, type the current time in the following format:

hrs:minutes:seconds

Zeros default for minutes and seconds if you leave these numbers blank

6. Press RETURN.
7. The DOS prompt is now displayed on the screen:

A>\_

### Bypassing date and time

1. Press RETURN twice.
2. The DOS prompt is now displayed on the screen:

A>\_

Note: Because the PCW-1 clock is operated with an automatic charged battery, there should be only two instances where you have to manually set the time. These are the dates where daylight savings time and standard time changes are made. PCW-1 calculates the day of week, correcting for leap years, for any date entered up to the year 2099 A.D.

### How to Change Disk Drives

Once you have the DOS prompt "A>" on the screen, DOS is available for operation. The prompt "A>" tells you that Drive A is ready or "active". This means that DOS assumes you will be working with the disk in the active drive.

There will be times, however, when you need to use the disk in Drive B, thereby making it the active or "current" drive. To **change** the current (active) drive, you simply type the letter "b" (upper or lower-case), followed by a colon, and press the RETURN key.

A>b:  
Press RETURN

Notice that the DOS prompt has changed from A> to B>.

B>\_

To **change back** to Drive A, type:

B>a:  
Press RETURN



Drive A is now active.

A>\_

### Exiting PCW-1 MS-DOS

#### **Exiting to the Word Processor**

If you wish to go directly from DOS to PCW-1 Word Processing, you can load or "warm boot" the PCW-1 System.

1. Display the DOS prompt A> on the screen.
2. Remove the DOS disk from Drive A.
3. Insert the PCW-1 System disk in Drive A.
4. Press ALT + SHIFT +  or  (CTRL + ALT + DEL on IBM keyboard) to get the "Welcome to PCW-1 Word Processing" message on the screen. You are now out of personal computing mode.

#### **Exiting to Power Off**

1. Display either DOS prompt A> or B> on the screen.
2. Turn off the power switch.

### Starting Application Software

Application software, such as Lotus 1-2-3, may or may not already contain DOS on its system disk. This information would be contained in the manual of the application software. Instructions for exiting to application software in both situations are discussed next.

#### **Starting application software without DOS on the system disk**

1. Display the DOS prompt "A>" on the screen.
2. Remove the DOS disk from Drive A.
3. Insert the selected software program disk in Drive A.
4. Enter the specific boot command as described in the documentation which accompanies the application software.

**Starting application software with DOS on the system disk**

1. Insert the selected software program disk in Drive A.
2. Turn on the system.

\*Note: Refer to the system documentation provided with the application software package for complete installation, start-up, and operating instructions.

## Files and Selected Commands

### File Types and Their Use

As discussed in Chapter 1, the PCW-1 MS-DOS operating system groups a number of instructions into a single program so the computer can carry out operations such as running the program, using the disk, and printing. All of these operating system instructions, plus application programs, data, and text are organized and stored on disk into different groupings known as files. There are basically three **types** of files: command files, application program files, and data files.

**Command Files** are just that--commands (instructions) for the computer to carry out. PCW-1 MS-DOS has two types of command files: internal commands and external commands. **Internal** commands can occur once PCW-1 MS-DOS is loaded and you type in the basic command at the keyboard. **External** commands are accessed when you type in the name of the file. PCW-1 MS-DOS then looks through the files on a disk, finds the one that has been requested, and loads it into memory.

**Application Program Files** are for a specific use, such as word processing, electronic spreadsheets (worksheets), and database management. An application program is stored within an external command file.

**Data Files**, usually referred to as **Text Files**, consist of all the information that you create when you use application programs. Text from word processing, numbers and formulas from spreadsheets (worksheets), and data from database management programs.

### How to Name MS-DOS Files

#### **Filenames**

A **filename** identifies a file on the disk. DOS requires that certain conventions, a standard format, be followed for giving names to files. You can choose any name you want for a file, as long as it is not exactly the same name as another file on the same disk.

The filename can consist of one to eight characters made up of letters, numbers, or a combination of both. A few other symbols are also allowed.

An example of a filename about the "1985 Business Plan" could be named:

'85BPLAN

### Extensions

You can also add an optional identification to the file, known as an "extension". The extension can be from one to three characters after the filename. You must include a period "." between the filename and the extension. Certain extensions are used to identify common types of files. They also keep your group-related files together.

For example, if you wanted a document file about the 1985 Business Plan, as well as a worksheet (spreadsheet) file relating to this business plan, you might name these two files as follows:

'85BPLAN.DOC (document file)

'85BPLAN.WKS (worksheet file)

Below is a list of common extensions and their meanings. You also have the option of developing your own extensions for other types of files you create.

#### Commonly-Used Extensions

.bas	BASIC language program
.bat	BATCH file
.com	standard program file
.doc	document file
.exe	standard program file
.lst	document file for printer
.tmp	temporary storage file
.wks	worksheet file (spreadsheet)

### Drive identifier

As you learned in **Basic Operation**, DOS works with the disk in the **active** drive. You will be at the DOS prompt A> after you have loaded DOS, until you change the disk drive to B. Therefore, any of the files you wish to work with will be on the disk in the currently active drive, unless you specify otherwise.

You can tell DOS to go to the other (inactive) drive simply by typing the letter of the drive followed by a colon. This is called the "drive identifier".

### **Filespec**

The sum of all of the standard parts--the filename, the extension, and the drive identifier--is called a "filespec".

Let's say Drive A is the active drive, and you wanted the 1985 Business Plan document file in Drive B, you would have to identify the file for DOS as follows:

```
A>b:'85BPLAN.DOC
```

You type the **filespec** in either upper or lower-case, without spaces between any of the parts.

At this point, you have learned about some basic operations, such as starting and exiting DOS, changing disk drives, types of files and their use, and naming files. The next step is to **use** the files. Copying and renaming files, formatting disks, viewing the directory (contents) of a disk, and other "housekeeping" activities are accomplished through PCW-1 **MS-DOS commands**.

The following pages of this section contain selected commands that are basic to working with files. Further details about DOS commands are contained in Section 2 of this manual, "Microsoft MS-DOS User's Guide."

### **How to Use MS-DOS Commands**

To request MS-DOS to perform an operation on a file, you type in the name of the command at the keyboard. In addition to a command, you must also provide DOS with the **filespec**, so that it knows the name and its extension, and the disk drive of the file. If the file is located on the currently active drive, it is not necessary to type in the drive identifier.



The general format for giving a command is:

**When the file is on the active drive**

```
A>command filename.extension
      or
B>command filename.extension
```

**When the file is on the inactive drive**

```
A>command b:filename.extension
      or
B>command a:filename.extension
```

[\*Be sure to type a space between the command and the filespec.]

Pressing the RETURN key causes the command to be "executed" or carried out.

Each of the selected commands that are described below are different functions, and depending on the type of function, there will be **variations** to this general format. Examples, specific formats, and the purpose of each command are provided for easy reference.

### Selected MS-DOS Commands

In this section, instructions pertain to either a **single** file or to **all** of the files on a disk. When using commands for **all** of the files on a disk, DOS recognizes the characters **\*.\*** to mean perform the specified command on all of the files. There will be situations when you need to work with **groups** of files, such as all of the files on a disk that contain the extension **.doc**. Please refer to the topic of "Wild Cards" in "Section 2, Microsoft MS-DOS User's Guide", of this manual.

When using any of the commands, pay special attention to how the entire command is typed, including the filespec; in particular, whether or not **spaces** are included or the **drive designator** is typed before or after the filename and extension.

**REMINDER:** The DOS prompt (either A> or B> ) indicates which drive is active.

**DIR (Directory)**

The **Directory** command displays a list of the files contained on a disk.

When you type in the command for directory, you use only the first three letters -- **DIR**

**Directory of ALL files - ACTIVE drive**

A>dir

Press RETURN

A sample directory looks like this.

```
A>dir
Volume in drive A has no label
Directory of A:\

COMMAND  COM
FORMAT   COM
CHKDSK   COM
MORE     COM
RECOVER  COM
SYS      COM
'85BPLAN DOC
ACCT     DOC
CLIENT   DOC
'85BPLAN WKS
BUDGET   WKS
```

**Directory of ALL files - INACTIVE drive**

A>dir b:

Press RETURN

When the list is too long to fit on the screen, select a wide screen display of the directory.

A>dir/w

Press RETURN

w means wide screen.

Directory of a SINGLE file - ACTIVE drive

A>dir filename.extension

PRESS RETURN

Directory of a SINGLE file - INACTIVE drive

A>dir b:filename.extension

PRESS RETURN

**DEL (Delete)**

Delete removes one or more files from a disk. Type only the first three letters for the command -- DEL

Deleting ALL files - ACTIVE drive

A>del \*.\*

Press RETURN

Deleting ALL files - INACTIVE drive

A>del b:\*.\*

Press RETURN

Deleting a SINGLE file - ACTIVE drive

A>del filename.ext

Press RETURN

Deleting a SINGLE file - INACTIVE drive

A>del b:filename.ext

Press RETURN

**COPY**

Copy places a copy of one or more files onto another disk.

If you are copying a single file to another disk , you have the option of using the same filename for the copy or changing the name when you use the Copy command.

If you wish, you can make a second copy of a file on the same disk. Since DOS does not allow using files with the same name on the same disk, you must change the name of the second copy of the file.

**Copying ALL files - ACTIVE drive to INACTIVE drive**

```
A>copy *.* b:
```

Press RETURN

**Copying a SINGLE file - ACTIVE drive to INACTIVE drive**

```
A>copy filename.extension b:
```

Press RETURN

**Copying a SINGLE file to the SAME disk**

```
A>copy filename.extension newfilename.extension
```

Press RETURN

**Example:**

```
A>copy '85BPLAN.DOC BPLAN'85.DOC.
```

Press RETURN

**DISKCOPY**

Diskcopy copies the **complete contents** of one disk onto another disk. The original disk remains intact. It is used for making backup copies of disks in the event that the original disk is lost or damaged.

The disk that you copy to **must** be a formatted disk. If the disk is new (unformatted), please refer to the instructions for **format** before attempting diskcopy.

The disk which is being copied is called the **SOURCE** disk. The disk serving as the back-up copy is called the **TARGET** disk. If the target disk has been used before and contains any files, those files will be erased and cannot be recovered.

Diskcopy ALL files - ACTIVE drive to INACTIVE drive

A>diskcopy a: b:

Press RETURN

**REN (Rename)**

This command changes the name of a single file or a group of files. (Please refer to "Using Commands with Groups of Files" for instructions about renaming groups of files.)

The renamed file remains on the same disk. You cannot rename a file on a disk in the inactive drive. Change drives if the file you wish to rename is in the inactive drive.

**Renaming a SINGLE file - ACTIVE drive**

A>REN filename.extension newfilename.extension

Press RETURN

**FORMAT**

This command formats new blank disks for their first use in PCW-1's personal computer mode.

This operation also erases all files on a used disk, and they cannot be recovered. Therefore, be sure that any disk you format has no useful files stored on it.

The disk to be formatted is placed in the inactive drive. The common procedure uses the DOS disk in Drive A and the disk to be formatted in Drive B.

When the disk has been formatted, the Format routine asks you if you wish to format more disks. Therefore, you can format several disks in one session if you select "y" (for yes). If you wish to format only one disk, select either "n" (for no) or press RETURN. You will be returned to the DOS prompt A>.

**Formatting a SINGLE disk - INACTIVE drive**

A>format b:

Press RETURN

The message "Insert new diskette for drive B: and strike any key when ready" appears on the screen. Insert the disk to be formatted in Drive B and press any key to begin the formatting process.

Wait for the message "Format complete

Format another (Y/N)?

To format another, TYPE: Y

The procedure and messages will be the same until you wish to exit this command.

To return to the DOS prompt, TYPE: N

**CAUTION: BE CAREFUL NOT TO FORMAT YOUR MS-DOS DISK! THIS WILL ONLY OCCUR IF YOU DO NOT SPECIFY THE DRIVE DESIGNATOR, B. PLACE A WRITE PROTECT TAB ON YOUR MS-DOS DISK TO PREVENT ACCIDENTAL FORMAT OR DELETION.**



**TYPE**

Type displays the contents of a file on the screen. This is an alternative to printing it on paper. You would use this command for displaying text files only. You can specify only a single file to be typed.

To display a SINGLE file on the screen - ACTIVE drive

A>type filename.extension

Press RETURN

To display a SINGLE file on the screen - INACTIVE drive

A>type b:filename.extension

Press RETURN

**Section 2**

**Microsoft MS-DOS User's Guide  
(plus PCW-1 supplement)**

## **System Requirements**

The MS-DOS for the PCW-1 disk is designed specifically for the Minolta Office System PCW-1.

## TABLE OF CONTENTS

### CHAPTER 1

#### INTRODUCTION

1.1	WHAT IS MS-DOS? . . . . .	1-2
1.2	WHAT IS AN OPERATING SYSTEM? . . . . .	1-2
1.3	WHY IS MS-DOS SO IMPORTANT? . . . . .	1-3
1.4	ABOUT THIS MANUAL . . . . .	1-3
1.5	SYNTAX NOTATION . . . . .	1-4

### CHAPTER 2

#### GETTING STARTED

2.1	WHAT HAPPENS WHEN YOU FIRST LOAD MS-DOS? . . . . .	2-2
2.2	HOW TO ENTER THE DATE AND TIME . . . . .	2-2
2.3	HOW TO CHANGE THE DEFAULT DRIVE . . . . .	2-4
2.4	HOW TO FORMAT YOUR DISKS . . . . .	2-4
2.4.1	The FORMAT Command . . . . .	2-5
2.5	HOW TO BACK UP YOUR DISKS . . . . .	2-6
2.5.1	The DISKCOPY Command . . . . .	2-6
2.6	AUTOMATIC PROGRAM EXECUTION . . . . .	2-8
2.7	FILES . . . . .	2-9
2.7.1	What Is A File? . . . . .	2-9
2.7.2	How MS-DOS Keeps Track Of Your Files . . . . .	2-9
2.7.3	The DIR (Show Directory) Command . . . . .	2-9
2.7.4	The CHKDSK (Check Disk) Command . . . . .	2-11
2.8	HOW TO TURN THE SYSTEM OFF . . . . .	2-11

### CHAPTER 3

#### MORE ABOUT FILES

3.1	HOW TO NAME YOUR FILES . . . . .	3-2
3.2	WILD CARDS . . . . .	3-3
3.2.1	The ? Wild Card . . . . .	3-3
3.2.2	The * Wild Card . . . . .	3-4
3.3	ILLEGAL FILENAMES . . . . .	3-5
3.4	HOW TO COPY YOUR FILES . . . . .	3-5
3.5	HOW TO PROTECT YOUR FILES . . . . .	3-7
3.6	DIRECTORIES . . . . .	3-7
3.7	FILENAMES AND PATHS . . . . .	3-10
3.7.1	Pathnames . . . . .	3-10
3.7.2	Pathing And External Commands . . . . .	3-11
3.7.3	Pathing And Internal Commands . . . . .	3-12
3.7.4	Displaying Your Working Directory . . . . .	3-13
3.7.5	Creating A Directory . . . . .	3-13
3.7.6	How To Change Your Working Directory . . . . .	3-14
3.7.7	How To Remove A Directory . . . . .	3-14

**CHAPTER 4 LEARNING ABOUT COMMANDS**

4.1	INTRODUCTION . . . . .	4-2
4.2	TYPES OF MS-DOS COMMANDS . . . . .	4-2
4.3	COMMAND OPTIONS . . . . .	4-3
4.4	INFORMATION COMMON TO ALL MS-DOS COMMANDS . . . . .	4-4
4.5	BATCH PROCESSING . . . . .	4-6
4.6	THE AUTOEXEC.BAT FILE . . . . .	4-8
4.6.1	How To Create An AUTOEXEC.BAT File . . . . .	4-11
4.7	CREATING A .BAT FILE WITH REPLACEABLE PARAMETERS . . . . .	4-12
4.7.1	Executing A .BAT File . . . . .	4-13
4.8	INPUT AND OUTPUT . . . . .	4-13
4.8.1	Redirecting Your Output . . . . .	4-14
4.8.2	Filters . . . . .	4-14
4.8.3	Command Piping . . . . .	4-15

**CHAPTER 5 COMMANDS**

5.1	COMMAND FORMATS . . . . .	5-2
5.2	COMMANDS . . . . .	5-2
	BACKUP . . . . .	5-5
	BREAK . . . . .	5-7
	CHDIR . . . . .	5-8
	CHKDSK . . . . .	5-9
	CLS . . . . .	5-14
	COPY . . . . .	5-15
	CTTY . . . . .	5-19
	DATE . . . . .	5-20
	DEL . . . . .	5-22
	DIR . . . . .	5-23
	DISKCOPY . . . . .	5-24
	EXE2BIN . . . . .	5-27
	EXIT . . . . .	5-30
	FIND . . . . .	5-31
	FORMAT . . . . .	5-33
	MKDIR . . . . .	5-35
	MORE . . . . .	5-36
	PATH . . . . .	5-37
	PRINT . . . . .	5-38
	PROMPT . . . . .	5-41
	RECOVER . . . . .	5-43
	REM . . . . .	5-44
	REN . . . . .	5-45
	RESTORE . . . . .	5-46
	RMDIR . . . . .	5-47
	SET . . . . .	5-48
	SORT . . . . .	5-49
	SYS . . . . .	5-51
	TIME . . . . .	5-53
	TYPE . . . . .	5-54
	VER . . . . .	5-55
	VERIFY . . . . .	5-56
	VOL . . . . .	5-57

5.3	BATCH PROCESSING COMMANDS . . . . .	5-58
	ECHO . . . . .	5-58
	FOR . . . . .	5-59
	GOTO . . . . .	5-60
	IF . . . . .	5-61
	PAUSE . . . . .	5-62
	SHIFT . . . . .	5-63
	ADDITIONAL COMMAND	
	TPM . . . . .	5-64

## CHAPTER 6 MS-DOS EDITING AND FUNCTION KEYS

6.1	SPECIAL MS-DOS EDITING KEYS . . . . .	6-2
6.2	CONTROL CHARACTER FUNCTIONS . . . . .	6-6

## CHAPTER 7 EDLIN

7.1	INTRODUCTION . . . . .	7-2
7.2	HOW TO START EDLIN . . . . .	7-2
7.3	SPECIAL EDITING KEYS . . . . .	7-3
7.4	COMMAND INFORMATION . . . . .	7-17
7.4.1	Command Options . . . . .	7-20
7.5	EDLIN COMMANDS . . . . .	7-21
7.6	ERROR MESSAGES . . . . .	7-47

## CHAPTER 8 FILE COMPARISON UTILITY (FC)

8.1	INTRODUCTION . . . . .	8-2
8.1.1	Limitations On Source Comparisons . . . . .	8-2
8.2	FILE SPECIFICATIONS . . . . .	8-2
8.3	HOW TO USE FC . . . . .	8-3
8.4	FC SWITCHES . . . . .	8-3
8.5	DIFFERENCE REPORTING . . . . .	8-5
8.6	REDIRECTING FC OUTPUT TO A FILE . . . . .	8-6
8.7	EXAMPLES . . . . .	8-6
8.8	ERROR MESSAGES . . . . .	8-10

## APPENDIX A INSTRUCTIONS FOR USERS WITH SINGLE-DRIVE SYSTEMS

## APPENDIX B DISK ERRORS

## APPENDIX C ANSI ESCAPE SEQUENCES

## APPENDIX D HOW TO CONFIGURE YOUR SYSTEM

## APPENDIX E MS-DOS MESSAGE DIRECTORY

## Index

**CHAPTER 1**  
**INTRODUCTION**

What Is MS-DOS?

What Is An Operating System?

Why Is MS-DOS So Important?

About This Manual

Syntax Notation

MS-DOS Files

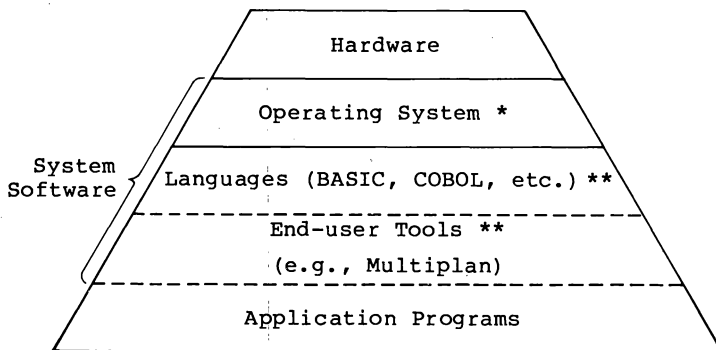
## 1.1 WHAT IS MS-DOS?

Microsoft(R) MS(tm)-DOS is a disk operating system for 8086/8088-based computers. Through MS-DOS, you communicate with the computer, disk drives, and printer, managing these resources to your advantage.

## 1.2 WHAT IS AN OPERATING SYSTEM?

An operating system is your "silent partner" when you are using the computer. It provides the interface between the hardware and both you (the user) and the other system software. An operating system can be compared to the electricity in a house--you need it for the toaster and the blender to work, but you are not always aware that it's there.

An operating system (OS) is the piece of system software most closely associated with the hardware. The OS is unique to the microprocessor (computer). For example, MS-DOS runs on the 8086/8088 microprocessor family and will not run on another microprocessor (like the Z8000) unless major parts of the OS are rewritten. Figure 1 illustrates how the hardware, the system software and the application software are related.



\* Must adapt to new hardware

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

Figure 1. Hardware/Software Relationships



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

### **1.3 WHY IS MS-DOS SO IMPORTANT?**

All Microsoft languages (BASIC Interpreter, BASIC Compiler, FORTRAN, COBOL, Pascal) are available under MS-DOS. Users of MS-DOS are assured that their operating system will be the first that Microsoft will support when any new products or major releases are announced.

### **1.4 ABOUT THIS MANUAL**

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

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

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

## 1.5 SYNTAX NOTATION

The following syntax notation is used throughout this manual in descriptions of command and statement syntax:

[ ] Square brackets indicate that the enclosed entry is optional.

< > Angle brackets indicate data you must enter. When the angle brackets enclose lower case text, you must type in an entry defined by the text; for example, <filename>. When the angle brackets enclose upper case text, you must press the key named by the text; for example, <RETURN>.

{ } Braces indicate that you have a choice between two or more entries. At least one of the entries enclosed in braces must be chosen unless the entries are also enclosed in square brackets.

... Ellipses indicate that an entry may be repeated as many times as needed or desired.

| A bar indicates an OR statement in a command. When used with an MS-DOS filter, the bar indicates a pipe.

**CAPS** Capital letters indicate portions of statements or commands that must be entered, exactly as shown.

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

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

**CHAPTER 2**  
**GETTING STARTED**

What Happens When You First Load MS-DOS?

How To Enter The Date And Time

How To Change The Default Drive

How To Format Your Disks

    The FORMAT Command

How To Back Up Your Disks

    The DISKCOPY Command

Automatic Program Execution

Files

    What Is A File?

    How MS-DOS Keeps Track Of Your Files

    The DIR (Show Directory) Command

    The CHKDSK (Check Disk) Command

How To Turn The System Off

## 2.1 WHAT HAPPENS WHEN YOU FIRST LOAD MS-DOS?

Follow your computer manufacturer's instructions to insert and load the MS-DOS disk into your system. Loading MS-DOS takes from 3 to 45 seconds, depending on the size of memory in your computer.

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

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

```
Current date is Wed 1-02-1981
Enter new date: _
```

You must now enter today's date and time at your terminal.

## 2.2 HOW TO ENTER THE DATE AND TIME

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

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

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

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

Any date is acceptable in answer to the new date prompt as long as it follows the above format. Separators between the numbers can be hyphens (-) or slashes (/). For example:

```
6-1-82 or 06/01/82
```

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

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

After you respond to the new date prompt and enter your

answer by pressing the <RETURN> key (or <ENTER> key on some terminals), you will see a prompt similar to this:

```
Current time is 8.30:14.32
Enter new time: _
```

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

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

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

MS-DOS uses this time value to keep track of when you last updated and/or created files on the system. Notice that MS-DOS uses 24-hour time; for instance, 1:30 p.m. is written 13:30. (See note on page 3-4, Section 1.)

Example:

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

You should only use the colon (:) to separate hours and minutes. If you enter an invalid number separator, MS-DOS will repeat the prompt.

#### NOTE

If you make a mistake while typing, press the control key on your keyboard, hold it down, and then press the C key. This <CONTROL-C> function will abort your current entry. You can then re-answer the prompt or type another command. To correct a line before you press <RETURN>, use the <BACKSPACE> key to erase one letter at a time.

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

### 2.3 HOW TO CHANGE THE DEFAULT DRIVE

After you have answered the new time prompt, a message is displayed that looks like this:

```
A>_
```

The A> is the MS-DOS prompt from the command processor. It tells you that MS-DOS is ready to accept commands. If you have inserted the MS-DOS disk into a drive other than A:, the command processor prompt will reflect that drive (for example, B>). However, usually you will load MS-DOS in drive A:.

The A in the previous prompt represents the default disk drive. This means that MS-DOS will search only the disk in drive A: for any filenames you may enter and will write files only to that disk unless you specify a different drive. You can ask MS-DOS to search the disk in drive B: by changing the drive designation or by specifying B: in a command. To change the disk drive designation, enter the new drive letter followed by a colon. For example:

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

The system prompt B> will appear and MS-DOS will search only the disk in drive B: until you specify a different default drive.

If you have only one disk drive attached to your computer, turn to Appendix A, "Instructions for Users with Single-Drive Systems," for important information. Your computer may have more than 2 disk drives; refer to your computer manufacturer's instructions on the use of these drives.

### 2.4 HOW TO FORMAT YOUR DISKS

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

A blank disk must be formatted with the MS-DOS FORMAT command. The FORMAT command changes the disk to a format that MS-DOS can use; it also analyzes the disk for defective tracks. If the disk is not already blank, formatting it will destroy any data that exists on the disk. Although this can be a convenient way to make a blank disk,

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

#### 2.4.1 The FORMAT Command

The syntax of the FORMAT command is:

```
FORMAT [d:]
```

where:

d: is the drive designation (the drive that contains the disk to be formatted)

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

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

```
FORMAT B:
```

MS-DOS issues the following message:

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

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

```
Formatting...
```

while MS-DOS is formatting your disk. If you add the /V switch to the FORMAT command, MS-DOS will ask for the volume label of the disk. (See below.) The /S switch tells MS-DOS to copy its system files onto the new disk.

(Note: this message may not appear on your screen.)

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

```
Formatting...Format complete  
System transferred
```

```
Volume label (11 characters. RETURN for none)?
```

Volume labels are useful to identify disks--they are like a



name tag for each disk. When you assign a unique volume label to a disk, you can always be sure that you know which disk you are using. The volume label you assign to a disk is displayed by issuing the MS-DOS VOL command (refer to Chapter 5, "MS-DOS Commands," for more information on the VOL command). Type a volume label in response to the above prompt if you want to identify this disk, and press <RETURN>. An example of a volume label is PROGRAMS. If you do not want to attach a label to this disk, simply press the <RETURN> key. You will see on your screen a message similar to this:

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

Format another (Y/N)?\_

Type Y to format another disk. Type N to end the FORMAT program.

## 2.5 HOW TO BACK UP YOUR DISKS

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

### 2.5.1 The DISKCOPY Command

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

The format of the DISKCOPY command is:

```
DISKCOPY [d:] [d:]
```

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

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

DISKCOPY A: B:

MS-DOS responds:

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

Make sure the MS-DOS disk is in drive A: and insert a blank, formatted disk in drive B: . Press any character key after you have done this and MS-DOS will begin copying the MS-DOS disk. After MS-DOS has copied the disk, MS-DOS displays:

Copy complete  
Copy another (Y/N)?

Type Y (for Yes) if you wish to copy another disk with DISKCOPY. If you type N (for No), the default drive prompt is displayed.

You now have a duplicate copy of your MS-DOS disk in drive B: . This duplicate copy can be saved as your backup copy of the MS-DOS disk.

Figure 2 illustrates the DISKCOPY command:

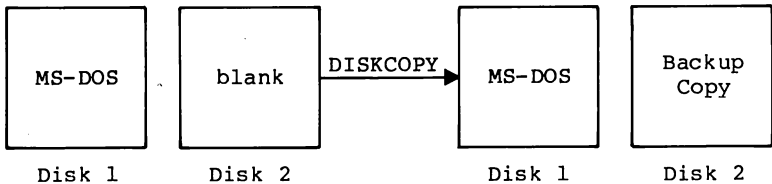


Figure 2. The DISKCOPY Command

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

#### NOTE

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

## 2.6 AUTOMATIC PROGRAM EXECUTION

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

When you start MS-DOS, the command processor searches for a file named AUTOEXEC.BAT on the MS-DOS disk. This file is a program that MS-DOS will run each time MS-DOS is started. Chapter 4, "Learning About Commands," tells you how to create an AUTOEXEC.BAT file.

## 2.7 FILES

### 2.7.1 What Is A File?

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

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

You create a file each time you enter and save data or text at your terminal. Files are also created when you write and name programs and save them on your disks.

### 2.7.2 How MS-DOS Keeps Track Of Your Files

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

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

These two system areas, the directories and the File Allocation Table, enable MS-DOS to recognize and organize the files on your disks. The File Allocation Table is copied onto a new disk when you format it with the MS-DOS FORMAT command and one empty directory is created, called the root directory.

### 2.7.3 The DIR (Show Directory) Command

If you want to know what files are on your disk, you can use the DIR command. This command tells MS-DOS to display all the files in the current directory on the disk that is named. For example, if your MS-DOS disk is in drive A: and you want to see the listing for the current directory on that disk, type:

## DIR A:

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

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

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

20 File(s)                    23040 bytes free

## NOTE

Two MS-DOS system files, IO.SYS and MSDOS.SYS, are "hidden" files and will not appear when you issue the DIR command.

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

```
DIR MYFILE.TXT
```

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

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

#### 2.7.4 The CHKDSK (Check Disk) Command

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

To check the disk in drive A:, type:

```
CHKDSK A:
```

MS-DOS will display a status report and any errors that it has found. An example of this display and more information on CHKDSK can be found in the description of the CHKDSK command in Chapter 5. You should run CHKDSK occasionally for each disk to ensure the integrity of your files.

#### 2.8 HOW TO TURN THE SYSTEM OFF

There is no "logoff" command in MS-DOS. To end your terminal session, open the disk drive doors and remove the disks. Then, simply turn your terminal off in response to a default drive prompt. Refer to your hardware manufacturer's instructions for information on how to do this.

#### NOTE

Always remove your disks from the disk drives before you turn off your terminal and disk drives.

## Summary of Commands in This Chapter

COMMAND	PURPOSE	SYNTAX
FORMAT	Formats disks for MS-DOS	FORMAT [d:]
DISKCOPY	Copies disks	DISKCOPY [drive1:][drive2:]
DIR	Lists directory information	DIR [d:][filename]
CHKDSK	Checks for errors on disk	CHKDSK [d:]

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

**CHAPTER 3**  
**MORE ABOUT FILES**

How To Name Your Files

Wild Cards

    The ? Wild Card

    The \* Wild Card

Illegal Filenames

How To Copy Your Files

How To Protect Your Files

Directories

    Filenames And Paths

        Pathnames

        Pathing And External Commands

        Pathing And Internal Commands

        Displaying Your Working Directory

        Creating A Directory

        How To Change Your Working Directory

        How To Remove A Directory



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

### 3.1 HOW TO NAME YOUR FILES

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

NEWFILE.EXE

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

A filename can be from 1 to 8 characters long. The filename extension can be three or fewer characters. You can type any filename in small or capital letters and MS-DOS will translate these letters into uppercase characters.

In addition to the filename and the filename extension, the name of your file may include a drive designation. A drive designation tells MS-DOS to look on the disk in the designated drive to find the filename typed. For example, to find directory information about the file NEWFILE.EXE which is located on the disk in drive A: (and drive A: is NOT the default drive), type the following command:

DIR A:NEWFILE.EXE

Directory information about the file NEWFILE.EXE is now displayed on your screen.

If drive A: is the default drive, MS-DOS will search only the disk in drive A: for the filename NEWFILE and so the drive designation is not necessary. A drive designation is needed if you want to tell MS-DOS to look on the other drive to find a file.

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

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

All of the parts of a filename comprise a file specification. The term file specification (or filespec) will be used in this manual to indicate the following filename format:

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

Remember that brackets indicate optional items. Angle brackets (<>) mean that you supply the text for the item. Note that the drive designation is not required unless you need to indicate to MS-DOS on which disk to search for a specific file. You do not have to give your filename a filename extension.

Examples of file specifications are:

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

## 3.2 WILD CARDS

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

### 3.2.1 The ? Wild Card

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

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

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

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

### 3.2.2 The \* Wild Card

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

```
DIR TEST*.EXE
```

will list all directory entries on the default drive with filenames that begin with the characters TEST and have an extension of .EXE. Here are some examples of files that might be listed by the above DIR command:

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

The wild card designation \*.\* refers to all files on the disk. Note that this can be very powerful and destructive when used in MS-DOS commands. For example, the command DEL \*.\* deletes all files on the default drive, regardless of filename or extension.

Examples:

To list the directory entries for all files named NEWFILE on drive A: (regardless of their filename extensions), simply type:

```
DIR A:NEWFILE.*
```

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

```
DIR B:?????????.TXT
```

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

### 3.3 ILLEGAL FILENAMES

MS-DOS treats some device names specially, and certain 3-letter names are reserved for the names of these devices. These 3-letter names cannot be used as filenames. They can be used as extensions. You must not name your files any of the following:

- AUX      Used when referring to input from or output to an auxiliary device (such as a printer or disk drive).
- CON      Used when referring to keyboard input or to output to the terminal console (screen).
- PRN      Used when referring to the printer device.
- NUL      Used when you do not want to create a particular file, but the command requires an input or output filename.

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

### 3.4 HOW TO COPY YOUR FILES

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

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

The format of the COPY command is:

```
COPY filespec [filespec]
```

For example:

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

will copy the file MYFILE.TXT on the disk in drive A: to a file named MYFILE.TXT on the disk in drive B:. A duplicate copy of MYFILE.TXT now exists.

Figure 3 illustrates how to copy files to another disk:

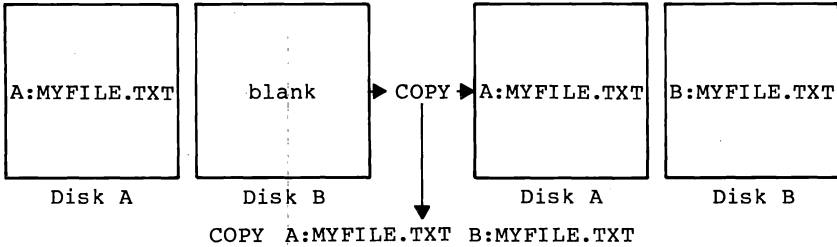


Figure 3. Copying Files to Another Disk

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

COPY A:MYFILE.TXT A:NEWNAME.TXT

You now have two copies of your file on disk A--one named MYFILE.TXT and the other named NEWNAME.TXT. The following figure illustrates this example:

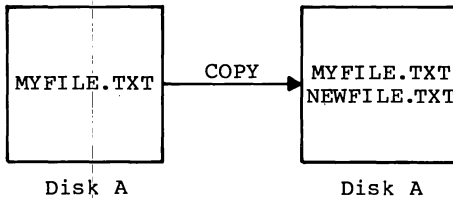


Figure 4. Copying Files on the Same Disk

You can also copy all files on a disk to another disk (i.e., make a backup copy) with the COPY command. Refer to Chapter 5, "MS-DOS Commands," for more information on this process.

### 3.5 HOW TO PROTECT YOUR FILES

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

### 3.6 DIRECTORIES

As you learned in Chapter 2, the names of your files are kept in a directory on each disk. The directory also contains information on the size of the files, their locations on the disk, and the dates that they were created and updated.

When there are multiple users on your computer, or when you are working on several different projects, the number of files in the directory can become large and unwieldy. You may want your own files kept separate from a co-worker's; or, you may want to organize your programs into categories that are convenient for you.

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

A hierarchical directory structure can be thought of as a "tree" structure: directories are branches of the tree and files are the leaves, except that the "tree" grows downward; that is, the "root" is at the top. The root is the first level in the directory structure. It is the directory that

is automatically created when you format a disk and start putting files in it. You can create additional directories and subdirectories by following the instructions in Chapter 4, "Learning About Commands."

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

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

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

Figure 5 illustrates a typical hierarchical directory structure:

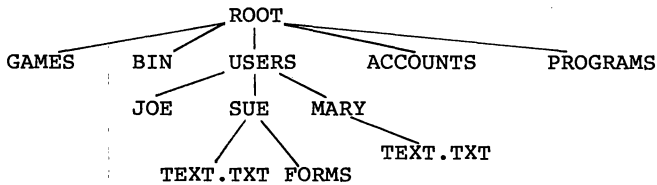


Figure 5. A Sample Hierarchical Directory Structure

The ROOT directory is the first level in the directory structure. You can create subdirectories from the ROOT by using the MKDIR command (refer to Chapter 5, "MS-DOS Commands," for information on MKDIR). In this example, five subdirectories of ROOT have been created. These include:

1. A directory of games, named GAMES
2. A directory of all external commands, named BIN (refer to Chapter 4, "Learning About Commands," for more information on the IN directory)
3. A USER directory containing separate subdirectories for all users of the system
4. A directory containing accounting information, named ACCOUNTS
5. A directory of programs, named PROGRAMS

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

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

```
DIR \USER\SUE\FORMS
```

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

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

```
DIR \USER\MARY
```



### 3.7 FILENAMES AND PATHS

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

#### 3.7.1 Pathnames

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

The syntax of pathnames is:

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

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

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

\	Indicates the root directory.
\PROGRAMS	Sample directory under the root directory containing program files.
\USER\MARY\FORMS\LA	A typical full pathname. This one happens to be a file named LA in the directory named FORMS belonging to the USER named MARY.

USER\SUE                   A relative pathname; it names the file or directory SUE in subdirectory USER of the working directory. If the working directory is the root directory (\), it names \USER\SUE.

TEXT.TXT                   Name of a file or directory in the working directory.

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

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

DIR ..

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

If you type:

DIR ..\..\

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

### 3.7.2 Pathing And External Commands

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

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

For example, if you are in a working directory named

\BIN\PROG, and all MS-DOS external commands are in \BIN, you must tell MS-DOS to choose the \BIN path to find the FORMAT command. The command:

```
PATH \BIN
```

tells MS-DOS to search in your working directory and the \BIN directory for all commands. You only have to specify this path once to MS-DOS during your terminal session. MS-DOS will now search in \BIN for the external commands. If you want to know what the current path is, type the word PATH and the current value of PATH will be printed.

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

### 3.7.3 Pathing And Internal Commands

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

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

The syntax of these four commands is shown below:

**COPY** <pathname> pathname>

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

**DEL** <pathname>

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

**DIR** <pathname>

Displays the directory for a specific path.

**TYPE** <pathname>

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

### 3.7.4 Displaying Your Working Directory

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

```
CHDIR<RETURN>
```

you will see:

```
A:\USER\JOE
```

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

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

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

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

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

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

### 3.7.5 Creating A Directory

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

### MKDIR NEWDIR

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

To put files in the new directory, use the MS-DOS line editor, EDLIN. Chapter 7, "The Line Editor (EDLIN)," describes how to use EDLIN to create and save files.

### 3.7.6 How To Change Your Working Directory

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

```
A>CHDIR \USER
```

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

### 3.7.7 How To Remove A Directory

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

```
RMDIR NEWDIR
```

Note that the directory NEWDIR must be empty except for the . and .. entries before it can be removed; this will prevent you from accidentally deleting files and directories. You can remove any directory by specifying its pathname. To remove the \BIN\USER\JOE directory, make sure that it has only the . and .. entries, then type:

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

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

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

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

#### Summary of Commands in This Chapter

COMMAND	PURPOSE	SYNTAX
COPY	Copies files	COPY <filespec> [<filespec>]
PATH	Sets MS-DOS search path	PATH [pathname]
CHDIR	Displays working directory; changes directories	CHDIR [pathname]
MKDIR	Makes a new directory	MKDIR [pathname]
RMDIR	Removes a directory	RMDIR [pathname]

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

**CHAPTER 4**  
**LEARNING ABOUT COMMANDS**

Introduction

Types Of MS-DOS Commands

Command Options

Information Common To All MS-DOS Commands

Batch Processing

The AUTOEXEC.BAT File

How to Create An AUTOEXEC.BAT File

Creating A .BAT File With Replaceable Parameters

Executing A .BAT File

Input And Output

Redirecting Your Output

Filters

Command Piping

#### 4.1 INTRODUCTION

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

Compare, copy, display, delete, and rename files.

Copy and format disks.

Execute system programs such as EDLIN, as well as your own programs.

Analyze and list directories.

Enter date, time, and remarks.

Set various printer and screen options.

Copy MS-DOS system files to another disk.

Request MS-DOS to wait for a specific period of time.

#### 4.2 TYPES OF MS-DOS COMMANDS

There are two types of MS-DOS commands:

Internal commands

External commands

Internal commands are the simplest, most commonly used commands. You cannot see these commands when you do a directory listing on your MS-DOS disk; they are part of the command processor. When you type these commands, they execute immediately. The following internal commands are described in Chapter 5:

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



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

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

When you enter an external command, do not include its filename extension. The following external commands are described in Chapter 5:

BACKUP	MORE
CHKDSK	PRINT
DISKCOPY	RECOVER
FIND	RESTORE
FORMAT	SORT
EXE2BIN	SYS

### 4.3 COMMAND OPTIONS

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

The following is the format of all MS-DOS commands:

Command [options...]

where [options...] represents the following:

d:	Refers to disk drive designation.
filename	Refers to any valid name for a disk file, including an optional filename extension. The filename option does <u>not</u> refer to a device or to a disk drive designation.
.ext	Refers to an optional filename extension consisting of a period and 1-3 characters. When used, filename extensions immediately follow filenames.

**filespec** Refers to an optional drive designation, a filename, and an optional three letter filename extension in the following format:

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

**pathname** Refers to a pathname or filename in the following format:

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

**switches** Switches are options that control MS-DOS commands. They are preceded by a forward slash (for example, /P).

**arguments** Provide more information to MS-DOS commands. You usually choose between arguments; for example, ON or OFF.

#### 4.4 INFORMATION COMMON TO ALL MS-DOS COMMANDS

The following information applies to all MS-DOS commands:

1. Commands are usually followed by one or more options.
2. Commands and options may be entered in uppercase or lowercase, or a combination of keys.
3. Commands and options must be separated by delimiters. Because they are easiest, you will usually use the space and comma as index delimiters. For example:

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

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

In this manual, we will use a space as the delimiter in commands.

4. Do not separate a file specification with delimiters, since the colon and the period already serve as delimiters.
5. When instructions say "Press any key," you can press any alpha (A-Z) or numeric (0-9) key.
6. You must include the filename extension when referring to a file that already has a filename extension.

7. You can abort commands when they are running by pressing <CONTROL-C>.
8. Commands take effect only after you have pressed the <RETURN> key.
9. Wild cards (global filename characters) and device names (for example, PRN or CON) are not allowed in the names of any commands.
10. When commands produce a large amount of output on the screen, the display will automatically scroll to the next screen. You can press <CONTROL-S> to suspend the display. Press any key to resume the display.
11. MS-DOS editing and function keys can be used when entering commands.  
Refer to Chapter 6, "MS-DOS Editing and Function Keys," for a complete description of these keys.
12. The prompt from the command processor is the default drive designation plus a greater-than sign; for example, A>.
13. Disk drives will be referred to as source drives and destination drives. A source drive is the drive you will be transferring information from. A destination drive is the drive you will be transferring information to.

#### 4.5 BATCH PROCESSING

Often you may find yourself typing the same sequence of commands over and over to perform some commonly used task. With MS-DOS, you can put the command sequence into a special file called a batch file, and execute the entire sequence simply by typing the name of the batch file. "Batches" of your commands in such files are processed as if they were typed at a terminal. Each batch file must be named with the .BAT extension, and is executed by typing the filename without its extension.

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

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

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

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

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

```
NEWDISK
```

The result is the same as if each of the lines in the .BAT file was entered at the terminal as individual commands.

Figure 6 illustrates the three steps used to write, save, and execute an MS-DOS batch file:

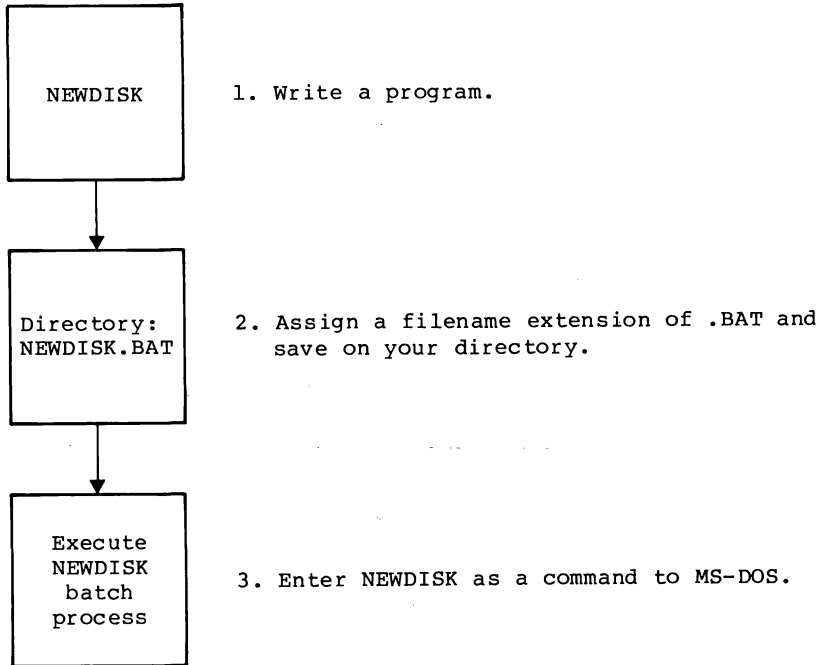


Figure 6. MS-DOS Batch File Steps

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

1. Do not enter the filename BATCH (unless the name of the file you want to execute is BATCH.BAT).
2. Only the filename should be entered to execute the batch file. Do not enter the filename extension.
3. The commands in the file named <filename>.BAT are executed.
4. If you press <CONTROL-C> while in batch mode, this prompt appears:

Terminate batch job (Y/N)?

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

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

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

#### 4.6 THE AUTOEXEC.BAT FILE

As discussed in Chapter 2, an AUTOEXEC.BAT file allows you to automatically execute programs when you start MS-DOS. Automatic Program Execution is useful when you want to run a specific package (for example, Microsoft Multiplan(tm)) under MS-DOS, and when you want MS-DOS to execute a batch program automatically each time you start the system. You can avoid loading two separate disks to perform either of these tasks by using an AUTOEXEC.BAT file:

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

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

If MS-DOS does not find an AUTOEXEC.BAT file when you first

load the MS-DOS disk, then the date and time prompts will be issued. Figure 7 illustrates how MS-DOS uses the AUTOEXEC.BAT file:

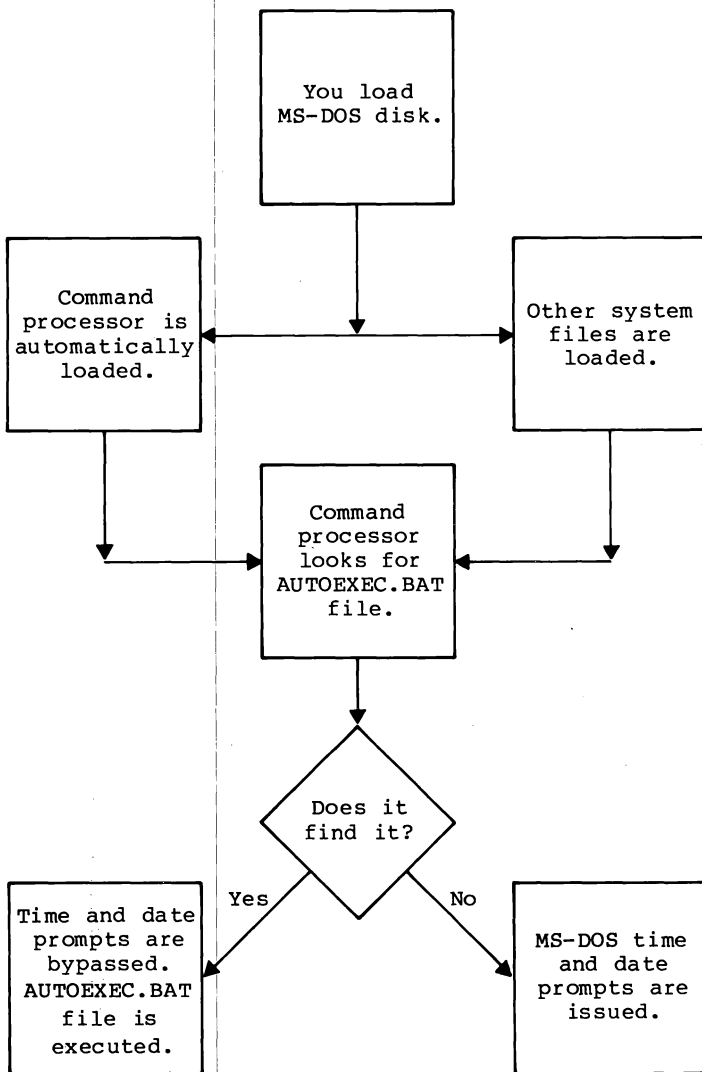


Figure 7. How MS-DOS Uses the AUTOEXEC.BAT File



#### 4.6.1 How To Create An AUTOEXEC.BAT File

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

1. Type:

```
COPY CON AUTOEXEC.BAT
```

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

2. Now type:

```
BASIC MENU
```

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

3. Press <CONTROL-Z>; then press the <RETURN> key to put the command BASIC MENU in the AUTOEXEC.BAT file.
4. The MENU program will now run automatically whenever you start MS-DOS.

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

#### NOTE

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

#### 4.7 CREATING A .BAT FILE WITH REPLACEABLE PARAMETERS

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

When used in MS-DOS commands, a parameter is an option that you define. With MS-DOS, you can create a batch (.BAT) file with dummy (replaceable) parameters. These parameters, named %0-%9, can be replaced by values supplied when the batch file executes.

For example, when you type the command line COPY CON MYFILE.BAT, the next lines you type are copied from the console to a file named MYFILE.BAT on the default drive:

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

Now, press <CONTROL-Z> and then press <RETURN>. MS-DOS responds with this message:

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

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

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

#### NOTES:

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

### 4.7.1 Executing A .BAT File

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

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

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

To execute the MYFILE batch process, type:

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

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

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

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

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

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

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

## 4.8 INPUT AND OUTPUT

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

### 4.8.1 Redirecting Your Output

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

```
DIR
```

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

```
DIR >MYFILES
```

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

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

```
DIR >>MYFILES
```

appends your directory listing to a currently existing file named MYFILES. If MYFILES does not exist, it is created.

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

```
SORT <NAMES >LIST1
```

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

### 4.8.2 Filters

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

MS-DOS filters include FIND, MORE, and SORT. Their

functions are described below:

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

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

### 4.8.3 Command Piping

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

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

```
DIR | SORT
```

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

Piping can also be used when you want to output to a file. If you want your directory sorted and sent to a new file (for example, DIREC.FIL), you could type:

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

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

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

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

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

```
DIR | SORT | MORE
```

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

You will find many uses for piping commands and filters.

#### Summary of Commands in This Chapter

COMMAND	PURPOSE	SYNTAX
REM	Adds comment line for batch files	REM [remark]
PAUSE	Suspends execution of a batch file	PAUSE [comment]
FIND	Searches for string of text	FIND string [filename]
MORE	Pages through a file 23 lines at a time	MORE
SORT	Sorts text	SORT

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

**CHAPTER 5**

**COMMANDS**

Command Formats

Commands

Batch Processing Commands

### 5.1 COMMAND FORMATS

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

1. You must enter any words shown in capital letters. These words are called keywords and must be entered exactly as shown. You can enter these keywords in any combination of upper/lowercase; MS-DOS will convert all keywords to uppercase.
2. You supply the text for any items enclosed in angle brackets (< >). For example, you should enter the name of your file when <filename> is shown in the format.
3. Items in square brackets ([ ]) are optional. If you wish to include optional information, do not include the square brackets, only the information within the brackets.
4. An ellipsis (...) indicates that you may repeat an item as many times as you want.
5. You must include all punctuation where shown (with the exception of square brackets), such as commas, equal signs, question marks, colons, or slashes.

### 5.2 COMMANDS

#### NOTE

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

The following MS-DOS commands are described in this chapter. Note that synonyms for commands are enclosed in parentheses.



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

CLS Clears screen

COPY Copies file(s) specified

CTTY Changes console TTY

DATE Displays and sets date

DEL Deletes file(s) specified (ERASE)

DIR Lists requested directory entries

DISKCOPY Copies disks

EXE2BIN Converts executable files to binary format

EXIT Exits command and returns to lower level

FIND Searches for a constant string of text

FORMAT Formats a disk to receive MS-DOS files

MKDIR Makes a directory (MD)

MORE Displays output one screen at a time

PATH Sets a command search path

PRINT Background print feature

PROMPT Designates command prompt

RECOVER Recovers a bad disk

REM Displays a comment in a batch file

REN Renames first file as second file (RENAME)

RESTORE Restores files

RMDIR Removes a directory (RD)

SET Sets one string value to another

SORT Sorts data alphabetically, forward or backward

SYS Transfers MS-DOS system files from drive A: to the drive specified

TIME Displays and sets time

TYPE Displays the contents of file specified

VER Prints MS-DOS version number

VERIFY Verifies writes to disk

VOL Prints volume identification number

## Batch Commands (Command extensions)

ECHO Turns batch file echo feature on/off

FOR Batch command extension

GOTO Batch command extension

IF Batch command extension

PAUSE Pauses for input in a batch file

SHIFT Increases number of replaceable parameters in batch process

NAME	TYPE
BACKUP	External

## PURPOSE

Backs up one or more files from a fixed disk to floppy disk.

## SYNTAX

```
BACKUP [<d:>][<path>][<filespec>] d: [/S][/M]
[/A][/P] [/D:<date>][/T:<time>][/L:fname]
```

## COMMENTS

The first parameter you specify is the fixed disk file(s) to back up. The second parameter is the backup disk drive. Unless otherwise specified, the old files on a backup floppy disk are erased before new files are added to it.

This backup program and the one supplied by IBM(r) are compatible. File and disk formats are the same as those used by the IBM backup program unless you set the /P switch or the /T switches described below. The files that are backed up with the /P switch are not guaranteed to be restorable by the IBM restore program.

The following switches are used with BACKUP:

- /S Back up subdirectories also.
- /M Only back up those files which have changed since the last backup.
- /A Add the files to be backed up to those already on the backup floppy disk. Do not erase old files on the floppy disk.
- /P Pack as many files as possible onto each floppy disk. Create a subdirectory on the floppy disk if that is the only way to fill the floppy disk. (WARNING: IBM Backup/Restore compatibility may be lost if this switch is used.)
- /D Only back up those files which were last modified at or after a certain date.
- /T Only back up those files which were last modified at or after a certain time.
- /L Make a backup log entry in the file specified. If no filename was given, the file BACKUP.LOG will be placed in the root directory of the files being backed up. The first line of the entry in the file will contain: [date time] where date and time are the backup dates and times. Each subsequent line in the entry will

correspond to one of the files that was backed up. These lines will consist of the file name and the number of the floppy disk that contains the file. This information can be used when you need to restore a particular file from a floppy disk. You will know exactly which disk to give RESTORE so that it will not have to search for files. If the backup log file already exists, the current entry is appended to the file.

The backup program sets the ERRORLEVEL in the following manner:

- 0 Normal completion
- 1 No files were found to back up
- 3 Terminated by user
- 4 Terminated due to error

NAME	TYPE
BREAK	Internal

## PURPOSE

Sets CONTROL-C check.

## SYNTAX

BREAK [ON|OFF]

## COMMENTS

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

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

NAME CHDIR (CHANGE DIRECTORY) TYPE Internal

SYNONYM  
CD

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

SYNTAX  
CHDIR [pathname]

## COMMENTS

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

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

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

```
CHDIR ..
```

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

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

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

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

NAME	TYPE
CHKDSK (CHECK DISK)	External

**PURPOSE**

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

**SYNTAX**

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

**COMMENTS**

CHKDSK should be run occasionally on each disk to check for errors in the directory. If any errors are found, CHKDSK will display error messages, if any, and then a status report.

A sample status report follows:

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

```
65536 bytes total memory
 53152 bytes free
```

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

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

```
CHKDSK A:>filename
```

The errors will be sent to the filename specified. Do not use the /F switch if you redirect CHKDSK output.

**MESSAGES**

<filename> contains non-contiguous blocks  
The file or files you named are not written contiguously on the disk.

All specified file(s) are contiguous  
The file or files you named are all written sequentially on disk.

The following errors will be corrected automatically if you specify the /F switch:

Cannot CHDIR to <filename>  
Tree past this point not processed  
CHKDSK is traveling the tree structure of the directory and is unable to proceed to the specified directory. All subdirectories underneath this directory will not be verified.

First cluster number is invalid  
entry truncated  
The file directory entry contains an invalid pointer to the data area. If you specified the /F switch, the file is truncated to a zero-length file.

Allocation error, size adjusted  
An invalid sector number was found in the FAT. The file was truncated at the end of the last valid sector.

Disk error reading FAT  
One of your File Allocation Tables has a defective sector in it. MS-DOS will automatically use the other FAT. It is a good idea to copy all your files onto another disk.

Disk error writing FAT  
One of your File Allocation Tables has a defective sector in it. MS-DOS will automatically use the other FAT. It is a good idea to copy all your files onto another disk.

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

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



**Insufficient memory**

Processing cannot continue

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

**Invalid drive specification**

Specify a valid drive.

**Invalid parameter**

One of the switches that you have specified is wrong.

**Invalid sub-directory entry**

The subdirectory that you have specified either does not exist or is invalid. Check to see that you have entered the subdirectory name correctly.

**Errors found, F parameter not specified**

Corrections will not be written to disk

You must specify the /F switch if you want the errors corrected by CHKDSK.

**Invalid current directory**

Processing cannot continue

Your disk is bad. Replace the disk or make a copy from your backup system disk.

**Cannot CHDIR to root**

Processing cannot continue

CHKDSK is traveling the tree structure of the directory and is unable to return to the root directory. CHKDSK is not able to continue checking the remaining subdirectories to the root.

**<filename> is cross linked on cluster**

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

**X lost clusters found in y chains**

Convert lost chains to files (Y/N)?

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

If you respond N to this prompt, and have specified the /F switch, CHKDSK will display:

X bytes disk space freed

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

X bytes disk space would be freed

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

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

Insufficient room in root directory  
Erase files in root and repeat CHKDSK  
CHKDSK cannot process until you delete files in the root directory.

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

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

If you respond N to this prompt, you may not be able to write to or read from the bad directory.

Entry has a bad attribute (or size or link)  
This message may be preceded by one or two periods to indicate which subdirectory is invalid. If you have specified the /F switch, CHKDSK will attempt to correct the error.

Cannot recover . entry, processing continued  
The . entry (current directory) is defective.

Cannot recover .. entry  
The .. (parent directory) entry is defective.

Directory is totally empty, no . or ..  
The specified directory does not contain references to current and parent directories. Delete the specified directory and recreate it.

(.) (..) does not exist

This is an informational message from  
CHKDSK. This message indicates either the  
. or the .. directory entry is invalid.

NAME	CLS	TYPE	Internal
------	-----	------	----------

PURPOSE Clears the terminal screen.

SYNTAX CLS

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

NAME	TYPE
COPY	Internal

**PURPOSE**

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

**SYNTAX**

COPY [<pathname>] [<pathname>] [/V] [A] [/B]  
(to copy)

COPY <filespec> + <filespec> ... <filespec>  
(to concatenate)

**COMMENTS** To copy files:**NOTE**

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 will be on the default drive and will have the same name as the original file (first pathname option). If the first pathname is on the default drive and the second pathname is not specified, the COPY will be aborted. (Copying files to themselves is not allowed.) MS-DOS will display the error message:

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

The second option may take three forms:

1. If the second option is a drive designation (d:) only, the original file is copied with the original filename to the designated drive.
2. If the second option is a filename only, the original file is copied to a file on the default drive with the filename specified.

3. If the second option is a full file specification, the original file is copied to a file on the default drive with the filename specified.

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

The /A and /B switches indicate that the files being processed are ASCII or binary files. Each switch applies to the file specification preceding it and to all remaining file specifications on the command line, until another /A or /B is encountered.

The following discussion applies to the A and /B switches:

When used with a source file specification:

- /A Causes the file to be treated as an ASCII (text) file. Data in the file is copied up to but not including the first end-of-file mark (in EDLIN, this is CTRL-Z). The remainder of the file is not copied.
- /B Causes the entire file to be copied, including any end-of-file mark.

When used with a target file specification:

- /A Causes an end-of-file character to be added as the last character of the file.
- /B Causes no end-of-file character to be added.

When you are concatenating files (see below), the default switch is always /A.

#### To concatenate files:

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

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

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

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

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

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

In the following example, for each file found matching \*.LST, that file is combined with the corresponding .REF file. The result is a file with the same filename but with the extension .PRN. Thus, FILE1.LST will be combined with FILE1.REF to form FILE1.PRN; then XYZ.LST with XYZ.REF to form XYZ.PRN; and so on:

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

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

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

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

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

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

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

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

This command appends all \*.LST files, except ALL.LST itself, to ALL.LST. This command will

not produce an error message and is the correct way to append files using the COPY command.

## MESSAGES

Cannot do binary reads from a device

The copy cannot be done in binary mode when you are copying from a device. Remove the /B switch or specify an ASCII copy with the /A switch.

Content of destination lost before copy

A file to be used as a source file to the COPY command has been overwritten prior to completion of the copy. Example: COPY A + B B destroys the file B before it can be copied.



NAME	CTTY	TYPE
		Internal

## PURPOSE

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

## SYNTAX

CTTY <device>

## COMMENTS

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

CTTY AUX

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

CTTY CON

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

NAME	TYPE
DATE	Internal

## PURPOSE

Enter or change the date known to the system. This date will be recorded in the directory for any files you create or alter.

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

## SYNTAX

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

## COMMENTS

If you type DATE, DATE will respond with the message:

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

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

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

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

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

The new date must be entered using numerals only; letters are not permitted. The allowed options are:

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

The date, month, and year entries may be separated by hyphens (-) or slashes (/). MS-DOS is programmed to change months and years correctly, whether the month has 31, 30, 29, or 28 days. MS-DOS handles leap years, too.

## MESSAGES

Invalid date

Enter new date:         

If the options or separators are not valid,  
DATE displays this message. Enter a valid  
date.

NAME                   DEL (DELETE)                   TYPE                   Internal

SYNONYM               ERASE

PURPOSE               Deletes all files with the designated file specification.

SYNTAX               DEL [pathname]

## COMMENTS

If the pathname is \*.\* , the prompt Are you sure? appears. If a Y or y is typed as a response, then all files are deleted as requested. You can also type ERASE for the DELETE command.

NAME	DIR (DIRECTORY)	TYPE	Internal
------	-----------------	------	----------

PURPOSE  
Lists the files in a directory.

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

#### COMMENTS

If you just type DIR, all directory entries on the default drive are listed. If only the drive specification is given (DIR d:), all entries on the disk in the specified drive are listed. If only a filename is entered with no extension (DIR filename), then all files with the designated filename on the disk in the default drive are listed. If you designate a file specification (for example, DIR d:filename.ext), all files with the filename specified on the disk in the drive specified are listed. In all cases, files are listed with their size in bytes and with the time and date of their last modification.

The wild card characters ? and \* (question mark and asterisk) may be used in the filename option. Note that for your convenience, the following DIR commands are equivalent:

COMMAND	EQUIVALENT
DIR	DIR *.*
DIR FILENAME	DIR FILENAME.*
DIR .EXT	DIR *.EXT

Two switches may be specified with DIR. The /P switch selects Page Mode. With /P, display of the directory pauses after the screen is filled. To resume display of output, press any key.

The /W switch selects Wide Display. With /W, only filenames are displayed, without other file information. Files are displayed five per line.

NAME DISKCOPY TYPE External

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

SYNTAX DISKCOPY [d:] [d:]

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

The disk in the destination drive must be formatted prior to using DISKCOPY.

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

After copying, DISKCOPY prompts:

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

If you press Y, the next copy is performed on the same drives that you originally specified, after you have been prompted to insert the proper disks.

To end the COPY, press N.

Notes:

1. If you omit both options, a single-drive copy operation will be performed on the default drive.
2. If you omit the second option, the default drive will be used as the destination drive.
3. Both disks must have the same number of physical sectors and those sectors must be the same size.

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

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

For example:

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

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

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

#### MESSAGES

Copy not completed  
DISKCOPY cannot copy the entire disk.

DISK error while reading drive A  
Abort, Ignore, Retry?

Disk errors have been encountered during DISKCOPY processing. Refer to Appendix B, "Disk Errors," for information on this message.

Disks must be the same size

You cannot copy the contents of a disk with a different format using DISKCOPY. Use the COPY command to copy files onto the disk.

Source and target diskettes are not the same format. Cannot do the copy  
You must have the same size and kind of disks to run DISKCOPY. Example: you cannot copy from a single-sided disk to a double-sided disk. Reformat the target disk to be of the same type as the source disk.



NAME	TYPE
EXE2BIN	External

**PURPOSE**

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

**SYNTAX**

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

**COMMENTS**

This command is useful only if you want to convert .EXE files to binary format. The file named by filespec is the input file. If no extension is specified, it defaults to .EXE. The input file is converted to .COM file format (memory image of the program) and placed in the output file. If you do not specify a drive, the drive of the input file will be used. If you do not specify an output filename, the input filename will be used. If you do not specify a filename extension in the output filename, the new file will be given an extension of .BIN.

The input file must be in valid .EXE format produced by the linker. The resident, or actual code and data part of the file must be less than 64K. There must be no STACK segment.

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

1. If CS:IP is not specified in the .EXE file, a pure binary conversion is assumed. If segment fixups are necessary (i.e., the program contains instructions requiring segment relocation), you will be prompted for the fixup value. This value is the absolute segment at which the program is to be loaded. The resulting program will be usable only when loaded at the absolute memory address specified by a user application. The command processor will not be capable of properly loading the program.

2. If CS:IP is specified as 0000:100H, it is assumed that the file is to be run as a .COM file with the location pointer set at 100H by the assembler statement ORG; the first 100H bytes of the file are deleted. No segment fixups are allowed, as .COM files must be segment relocatable; that is, they must assume the entry conditions explained in the Macro Assembler Manual. Once the conversion is complete, you may rename the resulting file with a .COM extension. Then the command processor will be able to load and execute the program in the same way as the .COM programs supplied on your MS-DOS disk.

## MESSAGES

- File cannot be converted  
CS:IP does not meet either of the criteria specified above, or it meets the .COM file criterion but has segment fixups. This message is also displayed if the file is not a valid executable file.
- File not found  
The file is not on the disk specified.
- Insufficient memory  
There is not enough memory to run EXE2BIN.
- File creation error  
EXE2BIN cannot create the output file. Run CHKDSK to determine if the directory is full, or if some other condition caused the error.
- Insufficient disk space  
There is not enough disk space to create a new file.
- Fixups needed - base segment (hex):  
The source (.EXE) file contained information indicating that a load segment is required for the file. Specify the absolute segment address at which the finished module is to be located.

File cannot be converted  
The input file is not in the correct  
format.

WARNING -Read error in EXE file.  
Amount read less than size in header  
This is a warning message only.

NAME	TYPE
EXIT	Internal

## PURPOSE

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

## SYNTAX

EXIT

## COMMENTS

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

A>COMMAND

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

NAME	TYPE
FIND	External

## PURPOSE

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

## SYNTAX

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

## COMMENTS

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

If no files are specified, FIND will take the input on the screen and display all lines that contain the specified string.

Switches for FIND are:

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

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

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

The string should be enclosed in quotes.  
Example:

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

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

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

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

## MESSAGES

Incorrect DOS version

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

FIND: Invalid number of parameters

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

FIND: Syntax error

You typed an illegal string when issuing the FIND command.

FIND: File not found <filename>

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

FIND: Read error in <filename>

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

FIND: Invalid parameter <option-name>

You specified an option that does not exist.

NAME	TYPE
FORMAT	External

## PURPOSE

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

## SYNTAX

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

## COMMENTS

This command initializes the directory and file allocation tables. If no drive is specified, the disk in the default drive is formatted.

The /O switch causes FORMAT to produce an IBM personal Computer DOS version 1.X compatible disk. The /O switch 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 IBM(r) PC DOS, as well as MS-DOS 1.25/2.00 and IBM PC DOS 2.00. This switch 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.

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

If the /S switch is specified, it must be the last switch typed; then FORMAT copies operating system files from the disk in the default drive to the newly formatted disk. The files are copied in the following order:

```
IO.SYS
MSDOS.SYS
COMMAND.COM
```

## MESSAGES

Disk unsuitable for system drive  
FORMAT detected a bad track on the disk where system files should reside. You should only store data on the disk you tried to format when this message was displayed.

Insufficient memory for system transfer  
Your memory configuration is insufficient  
to transfer the MS-DOS system files IO.SYS  
and MSDOS.SYS (the /S switch).

Invalid characters in volume label  
The volume label should contain only up to  
11 alphanumeric characters.

Track 0 bad - disk unusable  
FORMAT can accommodate for defective  
sectors on the disk except for those near  
the beginning. In this case, use another  
disk.



NAME	MKDIR	TYPE	Internal
------	-------	------	----------

SYNONYM  
MD

PURPOSE  
Makes a new directory.

SYNTAX  
MKDIR <pathname>

COMMENTS  
This command is used to create a hierarchical directory structure. When you are in your root directory, you can create subdirectories by using the MKDIR command. The command:

MKDIR \USER

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

MKDIR \USER\JOE

MESSAGES  
Unable to create directory  
MS-DOS could not create the directory you specified. Check to see that there is not a name conflict. You may have a file by the same name, or the disk may be full.

NAME MORE TYPE External

PURPOSE Sends output to console one screen at a time.

SYNTAX MORE

## COMMENTS

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

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

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

TYPE MYFILES.COM | MORE

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

NAME	PATH	TYPE
		Internal

## PURPOSE

Sets a command search path.

## SYNTAX

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

## COMMENTS

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

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

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

MS-DOS will now search the \BIN\USER\JOE directory for external commands until you set another path or shut down MS-DOS.

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

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

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

The command PATH with no options will print the current path. If you specify PATH ;, MS-DOS will set the NUL path, meaning that only the working directory will be searched for external commands.

## MESSAGES

No path

You typed PATH with no options but have not set a command search path.

NAME	TYPE
PRINT	External

## PURPOSE

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

## SYNTAX

PRINT [[filespec] [/T] [/C] [/P]]...

## COMMENTS

You will use the PRINT command only if you have a line printer attached to your computer. The following switches are provided with this command:

- /T TERMINATE: this switch deletes all files in the print queue (those waiting to be printed). A message to this effect will be printed.
- /C CANCEL: This switch turns on cancel mode. The preceding filespec and all following filespecs will be suspended in the print queue until you type a /P switch.
- /P PRINT: This switch turns on print mode. The preceding filespec and all following filespecs will be added to the print queue until you issue a /C switch.

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

## EXAMPLES:

PRINT /T                   Empties the print queue.

PRINT A:TEMP1.TST/C A:TEMP2.TST A:TEMP3.TST  
Removes the three files  
indicated from the print  
queue.

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

## MESSAGES

## All files canceled

If the /T (TERMINATE) switch is issued, the message "All files canceled by operator" will be output on your printer. If the current file being printed is canceled by a /C, the message "File canceled by operator" will be printed.

## Cannot open (filename)

Either MS-DOS cannot find the specified file to print or the file does not exist. Check the command for a valid filename.

Errors on list device indicate that it may be offline. Please check it  
Your printer is offline

## (filename) file not found

You switched disks when a file was queued up, but before it started to print. Reissue the PRINT command for that filename.

## List output is not assigned to a device

This message will be displayed if the "Name of list device" specified to the above prompt is invalid. Subsequent attempts will return the same message until a valid device is specified.

## Name of list device [PRN:]

This prompt appears when PRINT is run the first time. Any current device may be specified and that device then becomes the PRINT output device. As indicated in the brackets, simply pressing <RETURN> results in the device PRN being used.

## No files match d:XXXXXXXX.XXX

A filespec was given for files to add to the queue, but no files match a specification. NOTE: if there are no files in the queue to match the canceled filespec, no error message will appear.

## PRINT queue is empty

There are no files in the print queue.

**PRINT queue is full**

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

**Resident part of PRINT installed**

This is the first message that MS-DOS displays when you issue the PRINT command. It means that available memory has been reduced by several thousand bytes to process the PRINT command concurrent with other processes.

**Notes for PT-1 printer user**

1. Because this printer is not equipped with a paper end sensor please use continuous sheet paper for printing or limit your files to one page.
2. If you boot PCW-1 MS-DOS from the electronic typewriter via the CODE + V command, some page formats which were set on the electronic typewriter menu may alter DOS formats during the print process.
3. The PT-1 printer is capable of printing up to 11 inches per line. If a line of data exceeds 11 inches all excess characters will be overstruck at the right edge of the paper.

NAME	PROMPT	TYPE
	PROMPT	Internal

**PURPOSE** Changes the MS-DOS command prompt.

**SYNTAX** PROMPT [<prompt-text>]

**COMMENTS**

This command allows you to change the MS-DOS system prompt (for example, A>). If no text is typed, the prompt will be set to the default prompt, which is the default drive designation. You can set the prompt to a special prompt, such as the current time, by using the characters indicated below.

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

-----  
Specify  
This  
Character: To Get This Prompt:  
-----

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

## EXAMPLES:

```
PROMPT $n
Sets the default drive letter
prompt.
PROMPT Time = $t$ _Date = $d
Sets a two-line prompt which
prints:
Time = (current time)
Date = (current date)
```

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

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



NAME	TYPE
RECOVER	External

**PURPOSE**

Recovers a file or an entire disk containing bad sectors.

**SYNTAX**

RECOVER <filename | d:>

**COMMENTS**

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

To recover a particular file, type:

```
RECOVER <filename>
```

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

To recover a disk, type:

```
RECOVER <d:>
```

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

**MESSAGES**

(xxxx) of (xxxx) bytes recovered  
This message tells you the number of bytes that MS-DOS was able to recover from the disk

Warning - directory full

The root directory is too full for RECOVER processing. Delete some files in the root directory to free space.

## NAME

REM (REMARK)

## TYPE

Internal

## PURPOSE

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

## SYNTAX

REM [comment]

## COMMENTS

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

## EXAMPLE:

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

NAME		TYPE
REN (RENAME)		Internal

SYNONYM  
RENAME

PURPOSE  
Changes the name of the first option (filespec) to the second option (filename).

SYNTAX  
REN <pathname> <filename>

#### COMMENTS

The first option (filespec) must be given a drive designation if the disk resides in a drive other than the default drive. Any drive designation for the second option (filename) is ignored. The file will remain on the disk where it currently resides.

The wild card characters may be used in either option. All files matching the first filespec are renamed. If wild card characters appear in the second filename, corresponding character positions will not be changed.

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

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

In the next example, REN renames the file ABODE on drive B: to ADOBE:

```
REN B:ABODE ?D?B?
```

The file remains on drive B:.

#### MESSAGES

File not found

You tried to rename a filespec to a name already present in the directory.

NAME	TYPE
RESTORE	External

## PURPOSE

Restores files backed up using either the Microsoft or the IBM backup program.

## SYNTAX

```
RESTORE <d:> [<d:>][<path>][<filespec>][/S][/P]
[/B:<date>][/A:<date>][/E:<time>][/L:<time>]
[/M][/N]
```

## COMMENTS

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

This restore program and the one supplied by IBM are compatible except for the extra switches described below.

The following switches are used with RESTORE:

- /S - Restore subdirectories also.
- /P - If any hidden or read-only files match the file specification, prompt for permission to restore them.
- /B - Only restore those files which were last modified on or before the given date.
- /A - Only restore those files which were last modified on or after the given date.
- /E - Only restore those files which were last modified at or earlier than the given time.
- /L - Only restore those files which were last modified at or later than the given time.
- /M - Only restore those files which have been modified since the last backup.
- /N - Only restore those files which no longer exist on the destination disk.

The backup program sets the ERRORLEVEL in the following manner:

- 0 Normal completion
- 1 No files were found to back up
- 3 Terminated by user
- 4 Terminated due to error

NAME RMDIR (REMOVE DIRECTORY) TYPE Internal

SYNONYM RD

PURPOSE Removes a directory from a hierarchical directory structure.

SYNTAX RMDIR <pathname>

## COMMENTS

This command removes a directory that is empty except for the . and .. shorthand symbols. You must delete all files first.

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

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

The directory has been deleted from the directory structure.

NAME	TYPE
SET	Internal

## PURPOSE

Sets one string value equivalent to another string for use in later programs.

## SYNTAX

SET [<string=string>]

## COMMENTS

This command is meaningful only if you want to set values that will be used by programs you have written. An application program can check all values that have been set with the SET command by issuing SET with no options. For example, SET TTY=VT52 sets your TTY value to VT52 until you change it with another SET command.

The SET command can also be used in batch processing. In this way, you can define your replaceable parameters with names instead of numbers. If your batch file contains the statement "LINK %FILE%", you can set the name that MS-DOS will use for that variable with the SET command. The command SET FILE=DOMORE replaces the %FILE% parameter with the filename DOMORE. Therefore, you do not need to edit each batch file to change the replaceable parameter names. Note that when you use text (instead of numbers) as replaceable parameters, the name must be ended by a percent sign.

If you type SET with no arguments, MS-DOS displays the current setting of SET.

NAME	TYPE
SORT	External

## PURPOSE

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

## SYNTAX

SORT [/R] [/+n]

## COMMENTS

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

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

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

## EXAMPLES:

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

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

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

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

The command:

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

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





**Incompatible system size**

The system files IO.SYS and MSDOS.SYS do not take up the same amount of space on the destination disk as the new system will need.

NAME	TIME	TYPE
		Internal

**PURPOSE**  
Displays and sets the time.

**SYNTAX**  
TIME [<hh>[:<mm>]]

**COMMENTS**  
If the TIME command is entered without any arguments, the following message is displayed:

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

Press the <RETURN> key if you do not want to change the time shown. A new time may be given as an option to the TIME command as in:

```
TIME 8:20
```

The new time must be entered using numerals only; letters are not allowed. The allowed options are:

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

The hour and minute entries must be separated by colons. You do not have to type the <ss> (seconds) or <cc> (hundredths of seconds) options.

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

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

MS-DOS then waits for you to type a valid time.

## NAME

TYPE

## TYPE

Internal

## PURPOSE

Displays the contents of the file on the console screen.

## SYNTAX

TYPE &lt;filespec&gt;

## COMMENTS

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

NAME	VER	TYPE
		Internal

## PURPOSE

Prints MS-DOS version number.

## SYNTAX

VER

## COMMENTS

If you want to know what version of MS-DOS you are using, type VER. The version number will be displayed on your screen.

NAME	TYPE
VERIFY	Internal

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

**SYNTAX** VERIFY [ON|OFF]

**COMMENTS**

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

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

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

NAME	TYPE
VOL (VOLUME)	Internal

**PURPOSE** Displays disk volume label, if it exists.

**SYNTAX** VOL [d:]

**COMMENTS** This command prints the volume label of the disk in drive d:. If no drive is specified, MS-DOS prints the volume label of the disk in the default drive.

**MESSAGES** volume in drive x has no label  
The disk in the drive does not have a volume label.

### 5.3 BATCH PROCESSING COMMANDS

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

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

NAME	TYPE
ECHO	Internal

**PURPOSE**  
Turns batch echo feature on and off.

**SYNTAX**  
ECHO [ON |OFF| <message>]

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

If ON or OFF are not specified, the current setting is displayed.

ECHO <message> is only useful if ECHO is off and you are using a batch file. By typing ECHO and a message in your batch file, you can print messages on the console.



NAME	FOR	TYPE	Internal
------	-----	------	----------

## PURPOSE

Command extension used in batch and interactive file processing.

## SYNTAX

```
FOR %%<c> IN <set> DO <command>
(for batch processing)
FOR %<c> IN <set> DO <command>
(for interactive processing)
```

## COMMENTS

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

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

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

## Examples:

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

```
FOR %%f IN (FOO BAR BLECH) DO REM %%f
```

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

## MESSAGES

FOR cannot be nested  
Nesting of FOR statements is illegal.

NAME	TYPE
GOTO	Internal

## PURPOSE

Command extension used in batch file processing.

## SYNTAX

GOTO <label>

## COMMENTS

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

## Example:

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

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

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

NAME	IF	TYPE	Internal
------	----	------	----------

PURPOSE  
 Command extension used in batch file processing.

SYNTAX  
 IF <condition> <command>

COMMENTS  
 The parameter <condition> is one of the following:

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

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

EXIST <filename>  
 True if and only if <filename> exists.

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

The IF statement allows conditional execution of commands. When the <condition> is true, then the <command> is executed. Otherwise, the <command> is ignored.

EXAMPLES:

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

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

NAME	PAUSE	TYPE	Internal
------	-------	------	----------

PURPOSE      Suspends execution of the batch file.

SYNTAX       PAUSE [comment]

## COMMENTS

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

When the command processor encounters PAUSE, it prints:

    Strike a key when ready . . .

If you press <CONTROL-C>, another prompt will be displayed:

    Abort batch job (Y/N)?

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

The comment is optional and may be entered on the same line as PAUSE. You may also want to prompt the user of the batch file with some meaningful message when the batch file pauses. For example, you may want to change disks in one of the drives. An optional prompt message may be given in such cases. The comment prompt will be displayed before the "Strike a key" message.

NAME	SHIFT	TYPE
		Internal

## PURPOSE

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

## SYNTAX

SHIFT

## COMMENTS

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

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

then a SHIFT will result in the following:

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

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

NAME	TPM (TAILOR PCW MODE)	TYPE	External
------	-----------------------	------	----------

PURPOSE Tailors the RS232C port by causing the printer output to be routed to an Asynchronous Communication Adapter.

SYNTAX TPM [n[,baud[,parity[,databits[,stopbits[,p]]]]]]

where:

1. n (Communication Adapter Number)  
input range: either 1:, 2:, 3:, or 4:  
default: 1:
2. baud rate  
input range: either 110, 150, 300, 600,  
1200, 2400, 4800, or 9600  
default: 9600  
Note: only the first two characters are required; subsequent characters are ignored.
3. parity  
input range: either E(even), O(odd), or N(one)  
default: Even  
Note: input E, O, or N
4. databits  
input range: either 5, 6, 7, or 8  
default: 7 bits
5. stopbits  
input range: either 1 or 2  
default: 1 bit  
Notes: 1. When a baud rate is set at 110, the default of the baud rate will be set at 2.  
2. When databit is set at 5 and stopbit at 2, the stopbit is forced to be set at 1.5.
6. p(option)  
input range: p or none  
default: none  
Note: When P is entered, time out errors are continuously retried. To stop the defaults you can retry the loop by pressing Ctrl+Break

COMMENTS

1. Parameters can be input in any order.
2. Any parameter can be omitted, if the parameter defaults are accepted.

3. Always use a comma or space between parameters to keep them separated.
4. When you have typed more than two different types data for one parameter, the last one that was inputted is the one that is executed.

**EXAMPLES:**

TPM 2400,E,5,1 .....1: 2400, E, 5, 1

TPM E,5,6,2400,1 .....1: 2400, E, 6, 1

TPM 2: 24 0 8 2 .....2: 9600, O, 8, 2

## CHAPTER 6

### MS-DOS EDITING AND FUNCTION KEYS

Special MS-DOS Editing Keys

Control Character Functions



## 6.1 SPECIAL MS-DOS EDITING KEYS

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

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

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

The relationship between the command line and the template is shown in Figure 8:

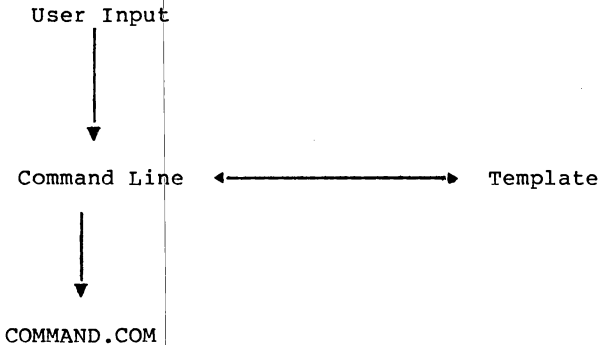


Figure 8. Command Line and Template

As seen in Figure 8, the user (you) types a command to MS-DOS on the command line. When you press the <RETURN> key, the command is automatically sent to the command processor (COMMAND.COM) for execution. At the same time, a copy of this command is sent to the template. You can now recall the command or modify it with MS-DOS special editing keys.

Table 6.1 contains a complete list of the special editing keys. Each of these keys is more fully described in Chapter 7, "The Line Editor (EDLIN)," where they can be used to edit your text files.

#### NOTE

The keys on your keyboard may not correspond to the specific keys in the following examples. Therefore, these MS-DOS editing keys will be referred to by FUNCTION rather than by name. When an example says to press the <SKIPl> key, find the key on your keyboard that corresponds to the "skip one character" editing function and press it. Some functions will require you to press two keys. Consult the operating manual for your terminal to determine which keys correspond to the MS-DOS editing functions described here.

You may wish to write in the keys on your keyboard that correspond to the editing keys described below. Space is provided in the following table to do this:

Table 6.1 Special Editing Functions

Key	Editing Function	Your Keyboard
<COPY1>	Copies one character from the template to the command line	
<COPYUP>	Copies characters up to the character specified in the template and puts these characters on the command line	
<COPYALL>	Copies all remaining characters in the template to the command line	
<SKIP1>	Skips over (does not copy) a character in the template	
<SKIPUP>	Skips over (does not copy) the characters in the template up to the character specified	
<VOID>	voids the current input; leaves the template unchanged	
<INSERT>	Enters/exits insert mode	
<NEWLINE>	Makes the new line the new template	
<CONTROL-Z	Puts a CONTROL-Z (LAH) end-of-file character in the new template	

## Examples:

If you type the following command:

```
DIR PROG.COM
```

MS-DOS displays information about the file PROG.COM on your screen. The command line is also saved in the template. To repeat the command, just press two keys: <COPYALL> and <RETURN>.

The repeated command is displayed on the screen as you type, as shown below:

```
<COPYALL>DIR PROG.COM<RETURN>
```

Notice that pressing the <COPYALL> key causes the contents

of the template to be copied to the command line; pressing <RETURN> causes the command line to be sent to the command processor for execution.

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

```
<COPYUP>C
```

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

```
DIR PROG._
```

Note that the underline is your cursor. Now type:

```
.ASM
```

The result is:

```
DIR PROG.ASM_
```

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

Now assume that you want to execute the following command:

```
TYPE PROG.ASM
```

To do this, type:

```
TYPE<INSERT> <COPYALL><RETURN>
```

Notice that when you are typing, the characters are entered directly into the command line and overwrite corresponding characters in the template. This automatic replacement feature is turned off when you press the insert key. Thus, the characters "TYPE" replace the characters "DIR " in the template. To insert a space between "TYPE" and "PROG.ASM", you pressed <INSERT> and then the space bar. Finally, to copy the rest of the template to the command line, you pressed <COPYALL> and then <RETURN>. The command TYPE PROG.ASM has been processed by MS-DOS, and the template becomes TYPE PROG.ASM.

If you had misspelled TYPE as BYTE, a command error would have occurred. Still, instead of throwing away the whole command, you could save the misspelled line before you press <RETURN> by creating a new template with the <NEWLINE> key:

```
BYTE PROG.ASM<NEWLINE>
```

You could then edit this erroneous command by typing:

T<COPY1>P<COPYALL>

The <COPY1> key copies a single character from the template to the command line. The resulting command line is then the command that you want:

TYPE PROG.ASM

As an alternative, you can use the same template containing BYTE PROG.ASM and then use the <SKIPl> and <INSERT> keys to achieve the same result:

<SKIPl><SKIPl><COPY1><INSERT>YP<COPYALL>

To illustrate how the command line is affected as you type, examine the keys typed on the left; their effect on the command line is shown on the right:

<SKIPl>	-	Skips over 1st template character
<SKIPl>	-	Skips over 2nd template character
<COPY1>	T	Copies 3rd template character
<INSERT>YP	TYP	Inserts two characters
<COPYALL>	TYPE PROG.ASM	Copies rest of template

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

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

## 6.2 CONTROL CHARACTER FUNCTIONS

A control character function is a function that affects the command line. You have already learned about <CONTROL-C> and <CONTROL-S>. Other control character functions are described below.

Remember that when you type a control character, such as <CONTROL-C>, you must hold down the control key and then press the C key.

Table 6.2 Control Character Functions

Control Character	Function
<CONTROL-N>	Toggles echoing of output to line printer.
<CONTROL-C>	Aborts current command.
<CONTROL-H>	Removes last character from command line and erases character from terminal screen.
<CONTROL-J>	Inserts physical end-of-line, but does not empty command line. Use the <LINE FEED> key to extend the current logical line beyond the physical limits of one terminal screen.
<CONTROL-P>	Toggles terminal output to line printer.
<CONTROL-S>	Suspends output display on terminal screen. Press any key to resume.
<CONTROL-X>	Cancel the current line; empties the command line; and then outputs a back slash (\), carriage return, and line feed. The template used by the special editing commands is not affected.

## **CHAPTER 7**

### **EDLIN**

Introduction

How to Start EDLIN

Special Editing Keys

Command Information

    Command Options

EDLIN Commands

Error Messages

## 7.1 INTRODUCTION

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

You can use EDLIN to:

1. Create new source files and save them.
2. Update existing files and save both the updated and original files.
3. Delete, edit, insert, and display lines.
4. 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.

## 7.2 HOW TO START EDLIN

To start EDLIN, type:

```
EDLIN <filespec>
```

If you are creating a new file, the <filespec> should be the name of the file you wish to create. If EDLIN does not find this file on a drive, EDLIN will create a new file with the name you specify. The following message and prompt will be 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, <filespec> should be the name of the file you want to edit. When EDLIN finds the file you specify on the designated or default drive, the file will be loaded into memory. If the entire file can be loaded, EDLIN will display 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 will load lines until memory is 3/4 full, then display 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 chapter for the procedure.

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 chapter 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 will not be 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 <filespec>.

### 7.3 SPECIAL EDITING KEYS

The special editing keys and template discussed in Chapter 6 can be used to edit your text files. These keys are discussed in detail in this section.

Table 7.1 summarizes the commands, codes, and functions. Descriptions of the special editing keys follow the table.

## NOTE

The keys on your keyboard may not correspond to the specific keys in the following examples. Therefore, these MS-DOS editing keys will be referred to by FUNCTION rather than by name. When an example says to press the <SKIPl> key, find the key on your keyboard that corresponds to the "skip one character" editing function and press it. Some functions may require you to press two keys. Consult the operating manual for your terminal to determine which keys correspond to the MS-DOS editing functions described here.

Table 7.1 Special Editing Keys

Function	Key	Description
Copy one character	<COPY1>	Copies one character from the template to the new line.
Copy up to character	<COPYUP>	Copies all characters from the template to the new line, up to the character specified.
Copy template	<COPYALL>	Copies all remaining characters in the template to the screen.
Skip one character	<SKIPl>	Does not copy (skips over) a character.
Skip up to character	<SKIPUP>	Does not copy (skips over) the characters in the template, up to the character specified.
Quit input	<VOID>	Voids the current input; leaves the template unchanged.
Insert mode	<INSERT>	Enters/exits insert mode.
Replace mode	<REPLACE>	Turns insert mode off; this is the default.
New template	<NEWLINE>	Makes the new line the new template.

## KEY

&lt;COPY1&gt;

## PURPOSE

Copies one character from the template to the command line.

## COMMENTS

Pressing the <COPY1> key copies one character from the template to the command line. When the <COPY1> key is pressed, one character is inserted in the command line and insert mode is automatically turned off.

## 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 <COPY1> key copies the first character (T) to the second of the two lines displayed:

```
1:*This is a sample file  
<COPY1> 1:*T_
```

Each time the <COPY1> key is pressed, one more character appears:

```
<COPY1> 1:*Th_  
<COPY1> 1:*Thi_  
<COPY1> 1:*This_
```

## KEY

&lt;COPYUP&gt;

## PURPOSE

Copies multiple characters up to a given character.

## COMMENTS

Pressing the <COPYUP> 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 <COPYUP>; it is not copied or displayed on the screen. Pressing the <COPYUP> 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 <COPYUP> also automatically turns off insert mode.

## EXAMPLE:

Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <COPYUP> key copies all characters up to the character specified immediately after the <COPYUP> key:

```
l:*This is a sample file  
<COPYUP>p l:*This is a sam_
```

## KEY

&lt;COPYALL&gt;

## PURPOSE

Copies template to command line.

## COMMENTS

Pressing the <COPYALL> key copies all remaining characters from the template to the command line. Regardless of the cursor position at the time the <COPYALL> key is pressed, the rest of the line appears, and the cursor is positioned after the last character on the line.

## EXAMPLE:

Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <COPYALL> key copies all characters from the template (shown in the upper line displayed) to the line with the cursor (the lower line displayed):

```
l:*This is a sample file (template)  
<COPYALL> l:*This is a sample file._ (command  
line)
```

Also, insert mode is automatically turned off.

## KEY

&lt;SKIPl&gt;

## PURPOSE

Skips over one character in the template.

## COMMENTS

Pressing the <SKIPl> key skips over one character in the template. Each time you press the <SKIPl> key, one character is not copied from the template. The action of the <SKIPl> key is similar to the <COPl> key, except that <SKIPl> skips a character in the template rather than copying it to the command line.

## EXAMPLE:

Assume that the screen shows:

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

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <SKIPl> key skips over the first character (T):

```
l:*This is a sample file  
<SKIPl> l:*_
```

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 <COPl> key, which moves the cursor beyond the last character of the line:

```
l:*This is a sample file.  
<SKIPl> l:*_  
<COPl> l:*his is a sample file._
```

## KEY

&lt;SKIPUP&gt;

## PURPOSE

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

## COMMENTS

Pressing the <SKIPUP> 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 <SKIPUP> key is similar to the <COPYUP> key, except that <SKIPUP> 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 <SKIPUP> key skips over all the characters in the template up to the character pressed after the <SKIPUP> key:

```
1:*This is a sample file
<SKIPUP>p 1:*_
```

The cursor position does not change. To see how much of the line has been skipped over, press the <COPYALL> key to copy the template. This moves the cursor beyond the last character of the line:

```
1:*This is a sample file:
<SKIPUP>p<COPYALL> 1:*ple file._
```



## KEY

&lt;VOID&gt;

## PURPOSE

Quits input and empties the command line.

## COMMENTS

Pressing the <VOID> key empties the command line, but it leaves the template unchanged. <VOID> also prints a back slash (\), 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 <COPYALL> key copies the template to the command line and the command line appears as it was before <VOID> was pressed.

## 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 <VOID>. Notice that a backslash appears on the Sample File line to tell you it has been cancelled:

```
1:*This is a sample file.  
<VOID> 1:*Sample File\  
1: _
```

Press <RETURN> to keep the original line, or to perform any other editing functions. If <COPYALL> is pressed, the original template is copied to the command line:

```
<COPYALL> 1: This is a sample file._
```

## KEY

&lt;INSERT&gt;

## PURPOSE

Enters/exits insert mode.

## COMMENTS

Pressing the <INSERT> key causes EDLIN to enter and exit insert mode. The current cursor position in the template is not changed. The cursor does move as each character is inserted. However, when you have finished inserting characters, the cursor will be positioned at the same character as it was before the insertion began. Thus, characters are inserted in front of the character to which the cursor points.

## EXAMPLE:

Assume that the screen shows:

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

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 <COPYUP> and f keys:

```
l:*This is a sample file
<COPYUP>f l:*This is a sample _
```

Now press the <INSERT> key and insert the characters "edit" and a space:

```
l:*This is a sample file.
<COPYUP>f l:*This is a sample _
<INSERT>edit l:*This is a sample edit _
```

If you now press the <COPYALL> key, the rest of the template is copied to the line:

```
l:*This is a sample edit
<COPYALL> l:*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:

```
<INSERT>edit <RETURN> 1:*This is a sample edit _
```

To exit insert mode, simply press the <INSERT> key again.

## KEY

&lt;REPLACE&gt;

## PURPOSE

Enters replace mode.

## COMMENTS

Pressing the <REPLACE> key causes EDLIN to exit insert mode and to enter replace mode. All the characters you type will overstrike and 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 will be 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 <COPYUP>m, <INSERT>lary, <REPLACE> tax, and then <COPYALL>:

```
1:*This is a sample file.
<COPYUP>m 1:*This is a sa_
<INSERT>lary 1:*This is a salary_
<REPLACE> tax 1:*This is a salary tax_
<COPYALL> 1:*This is a salary tax file._
```

Notice that you inserted lary and replaced mple with tax. If you type characters that extend beyond the length of the template, the remaining characters in the template will be automatically appended when you press <COPYALL>.

## KEY

&lt;NEWLINE&gt;

## PURPOSE

Creates a new template.

## COMMENTS

Pressing the <NEWLINE> key copies the current command line to the template. The contents of the old template are deleted. Pressing <NEWLINE> 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

<NEWLINE> performs the same function as the <VOID> key, except that the template is changed and an @ ("at sign" character) is printed instead of a \ (backslash).

## EXAMPLE:

Assume that the screen shows:

```
l:*This is a sample file.
l;*_
```

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 <COPYUP>m, <INSERT>lary, <REPLACE> tax, and then <COPYALL>:

```
l:*This is a sample file.
<COPYUP>m l:*This is a sa_
<INSERT>lary l:*This is a salary_
<REPLACE> tax l:*This is a salary tax_
<COPYALL> l:*This is a salary tax file._
```

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

<NEWLINE>l:\*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.

#### 7.4 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 will allow 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 <CONTROL-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<CONTROL-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 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<CONTROL-V>Z  
will find the first occurrence  
of CONTROL-Z in a file

R<CONTROL-V>Z<CONTROL-Z>foo  
will replace all occurrences  
of CONTROL-Z in a file by foo

S<CONTROL-V>C<CONTROL-Z>bar  
will replace 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 switch to tell EDLIN to ignore any CONTROL-Z characters in the file and to show you the entire file.



The EDLIN commands are summarized in the following table. They are also described in further detail in Section 7.5.

Table 7.2 EDLIN Commands

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

### 7.4.1 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 <CONTROL-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.

## 7.5 EDLIN COMMANDS

The following pages describe EDLIN editing commands.

## NAME

Append

## 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

[&lt;n&gt;]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 will 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 chapter for information on how to write edited lines to disk.

## NOTES

1. If you do not specify the number of lines to append, lines will be appended to memory until available memory is 3/4 full. No action will be 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.

## NAME

Copy

## 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> are omitted, the default is the current line. The file is renumbered automatically after the copy.

The line numbers must not overlap or you will get an "Entry error" message. 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.

## NAME

Delete

## PURPOSE

Deletes a specified range of lines in a file.

## SYNTAX

[&lt;line&gt;][,&lt;line&gt;]D

## COMMENTS

If the first <line> is omitted, that option will default 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> will be 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

```
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.



## NAME

Edit

## PURPOSE

Edits line of text.

## SYNTAX

[&lt;line&gt;]

## 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.

If no line number is typed (that is, if only the <RETURN> key is pressed), the line after the current line (marked with an asterisk (\*)) is edited. If no changes to the current line are needed and the cursor is at the beginning or end of the line, press the <RETURN> key to accept the line as is.

## 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 <COPYALL> special editing key,  
type:

```
<INSERT>number      4: number_  
<COPYALL><RETURN>  4: number_four.  
5:* _
```

## NAME

End

## 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. Therefore, you cannot tell EDLIN on which drive to save the file. The drive you want to save the file on must be selected when the editing session is started. If the drive is not selected when EDLIN is started, the file will be saved on the disk in the default drive. It will still be 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 will be aborted and the edited file lost, although part of the file might be written out to the disk.

## EXAMPLE:

E&lt;RETURN&gt;

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

## NAME

Insert

## PURPOSE

Inserts text immediately before the specified <line>.

## SYNTAX

[&lt;line&gt;]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 <CONTROL-C> is typed. When the insert is completed and insert mode has been exited, 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 will be the current line number and the lines will be 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 will be appended to the end of the file. In this case, the last line inserted will become 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:

7I

The result is:

7: \_

Now, type the new text for line 7:

7: and renumber lines

Then to end the insertion, press <CONTROL-Z> on the next line:

8: <CONTROL-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 <CONTROL-Z> on the next line:

8: so they are consecutive
9: <CONTROL-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 <CONTROL-Z> on  
line 14. The new lines will 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.

## NAME

List

## PURPOSE

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

## SYNTAX

[&lt;line&gt;][,&lt;line&gt;]L

## COMMENTS

Default values are provided if either one or both of the options are omitted. If you omit the first option, as in:

,&lt;line&gt;L

the display will start 11 lines before the current line and end 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 will be the same as if you omitted both options.

If you omit the second option, as in:

&lt;line&gt;L

23 lines will be displayed, starting with the specified <line>.

If you omit both parameters, as in:

L

23 lines will be 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 will be 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:

```
,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

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.

## NAME

Move

## PURPOSE

Moves a range of text to the line specified.

## SYNTAX

[&lt;line&gt;],[&lt;line&gt;],&lt;line&gt;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. 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 will display an "Entry error" message.

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

```
20,30,100M
```

## NAME

Page

## PURPOSE

Pages through a file 23 lines at a time.

## SYNTAX

[<line>][,<line>]P

## COMMENTS

If the first <line> is omitted, that number will default to the current line plus one. If the second <line> is omitted, 23 lines will be listed. The new current line becomes the last line displayed and is marked with an asterisk.

## NAME

Quit

## 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 chapter 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 will no longer exist.

## EXAMPLE:

```
Q
Abort edit (Y/N)?Y<RETURN>
A>_
```

## NAME

Replace

## PURPOSE

Replaces all occurrences of a string of text in the specified range with a different string of text or blanks.

## SYNTAX

```
[<line>][,<line>][?]R<string1><CONTROL-Z>
<string2>
```

## COMMENTS

As each occurrence of <string1> is found, it is replaced by <string2>. Each line in which a replacement occurs will be displayed. If a line contains two or more replacements of <string1> with <string2>, then the line will be 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 <CONTROL-Z>. <String2> must also be ended with a <CONTROL-Z><RETURN> combination or with a simple <RETURN>.

If <string1> is omitted, then Replace will take the old <string1> as its value. If there is no old <string1>, i.e., this is the first replace done, then the replacement process will be terminated immediately. If <string2> is omitted, then <string1> may be ended with a <RETURN>. If the first <line> is omitted in the range argument (as in ,<line>) then the first <line> will default to the line after the current line. If the second <line> is omitted (as in <line> or <line>,) , the second <line> will default to #. Therefore, this is the same as <line>,#. Remember that # indicates the line after the last line of the file.

If <string1> is ended with a <CONTROL-Z> and there is no <string2>, <string2> will be taken as an empty string and will become the new replace string. For example:

```
R<string2><CONTROL-Z><RETURN>
```

will delete occurrences of <string1>, but

R<string1><return> and  
R<RETURN>

will replace <string1> by the old <string2> and the old <string1> with the old <string2>, respectively. Note that "old" here refers to a previous string specified either in a Search or a Replace command.

If the question mark (?) option is given, the Replace command will stop at each line with a string that matches <string1>, display the line with <string2> in place, and then display the prompt O.K.?. If you press Y or the <RETURN> key, then <string2> will replace <string1>, and the next occurrence of <string1> will be found. Again, the O.K.? prompt will be displayed. This process will continue 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, the <string1> will be left as it was in the line, and Replace will go to the next occurrence of <string1>. If <string1> occurs more than once in a line, each occurrence of <string1> will be replaced individually, and the O.K.? prompt will be displayed after each replacement. In this way, only the desired <string1> will be 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<CONTROL-Z>or<RETURN>

The result is:

4: Delete or Insert  
 5: (the D or I commors)  
 8: The insert commor can place new lines

Note that in the above replacement, 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<CONTROL-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  
 .  
 .

## NAME

Search

## 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 will terminate when a match is found. If no line contains a match for <string>, the message "Not found" will be displayed.

If the question mark option (?) is included in the command, EDLIN will display the first line with a matching string; it will then prompt you with the message O.K.?. If you press either the Y or <RETURN> key, the line will become the current line and the search will terminate. If you press any other key, the search will continue 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 ,<line> S<string>), the first <line> will default 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> will default to # (line after last line of file), which is the same as <line>,# S<string>. If <string> is omitted, Search will take 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 will terminate 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:

```
2,12 Sand<RETURN>
```

The following line is displayed:

```
4: Delete and Insert
```

To get the "and" in line 5, modify the search command by typing:

```
<SKIPl><COPYALL>,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 without the verification (O.K.), type:

SXYZ

EDLIN will report a match and will continue to search for the same string when you issue the S command:

S

EDLIN reports another match.

S

EDLIN reports the string is not found.

Note that <string> defaults to any string specified by a previous Replace or Search command.

## NAME

Transfer

## PURPOSE

Inserts (merges) the contents of <filename> into the file currently being edited at <line>. If <line> is omitted, then the current line will be used.

## SYNTAX

[&lt;line&gt;]T&lt;filename&gt;

## 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.

## NAME

Write

## PURPOSE

Writes a specified number of lines to disk from the lines that are being edited in memory. Lines are written to disk beginning with line number 1.

## SYNTAX

[&lt;n&gt;]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 will be written until memory is 3/4 full. No action will be taken if available memory is already more than 3/4 full. All lines are renumbered, so that the first remaining line becomes line number 1.

## 7.6 ERROR MESSAGES

When EDLIN finds an error, one of the following error messages is displayed:

### Cannot edit .BAK file--rename file

**Cause:** 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.

**Cure:** If you need the .BAK file for editing purposes, you must either **RENAME** the file with a different extension; or **COPY** the .BAK file and give it a different filename extension.

### No room in directory for file

**Cause:** When you attempted to create a new file, either the file directory was full or you specified an illegal disk drive or an illegal filename.

**Cure:** Check the command line that started EDLIN for illegal filename and illegal disk drive entries. If the command is no longer on the screen and if you have not yet typed a new command, the EDLIN start command can be recovered by pressing the <COPYALL> 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

**Cause:** The last command typed contained a syntax error.

**Cure:** Retype the command with the correct syntax and press <RETURN>.

## Line too long

Cause: During a Replace command, the string given as the replacement caused the line to expand beyond the limit of 253 characters. EDLIN aborted the Replace command.

Cure: Divide the long line into two lines, then try the Replace command twice.

## Disk Full--file write not completed

Cause: You gave the End command, but the disk did not contain enough free space for the whole file. EDLIN aborted the E command and returned you to the operating system. Some of the file may have been written to the disk.

Cure: 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 will not be 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

Cause: You attempted to run EDLIN under a version of MS-DOS that was not 2.0 or higher.

Cure: You must make sure that the version of MS-DOS that you are using is 2.0 or higher.

## Invalid drive name or file

Cause: You have not specified a valid drive or filename when starting EDLIN.

Cure: Specify the correct drive or filename.

## Filename must be specified

Cause: You did not specify a filename when you started EDLIN.

Cure: Specify a filename.

## Invalid Parameter

Cause: You specified a switch other than /B when starting EDLIN.

Cure: Specify the /B switch when you start EDLIN.

## Insufficient memory

Cause: There is not enough memory to run EDLIN.

Cure: You must free some memory by writing files to disk or by deleting files before restarting EDLIN. Type a Write command and then an Append command to free memory.

## File not found

Cause: The filename specified during a Transfer command was not found.

Cure: Specify a valid filename when issuing a Transfer command.

## Must specify destination number

Cause: A destination line number was not specified for a Copy or Move command.

Cure: Reissue the command with a destination line number.

## Not enough room to merge the entire file

Cause: There was not enough room in memory to hold the file during a Transfer command.

Cure: You must free some memory by writing some files to disk or by deleting some files before you can transfer this file.

## File creation error

Cause: The EDLIN temporary file cannot be created.

Cure: Check to make sure that the directory has enough space to create the temporary file. Also, make sure that the file does not have the same name as a subdirectory in the directory where the file to be edited is located.

## CHAPTER 8

### FILE COMPARISON UTILITY (FC)

Introduction

Limitations On Source Comparisons

File Specifications

How To Use FC

FC Switches

Difference Reporting

Redirecting FC Output To A File

Examples

Error Messages



## 8.1 INTRODUCTION

It is sometimes useful to compare files on your disk. If you have copied a file and later want to compare copies to see which one is current, you can use the MS-DOS File Comparison Utility (FC).

The File Comparison Utility compares the contents of two files. The differences between the two files can be output to the console or to a third file. The files being compared may be either source files (files containing source statements of a programming language); or binary (files output by the MACRO-86 assembler, the MS-LINK Linker utility, or by a Microsoft high-level language compiler).

The comparisons are made in one of two ways: on a line-by-line or a byte-by-byte basis. The line-by-line comparison isolates blocks of lines that are different between the two files and prints those blocks of lines. The byte-by-byte comparison displays the bytes that are different between the two files.

### 8.1.1 Limitations On Source Comparisons

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

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

For binary files larger than available memory, FC compares both files completely, overlaying the portion in memory with the next portion from disk. All differences are output in the same manner as those files that fit completely in memory.

## 8.2 FILE SPECIFICATIONS

All file specifications use the following syntax:

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

where: d: is the letter designating a disk drive. If the drive designation is omitted, FC defaults to the operating system's (current) default drive.

filename is a one- to eight-character name of the file.

.ext is a one- to three-character extension to the filename.

### 8.3 HOW TO USE FC

The syntax of FC is as follows:

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

FC matches the first file (filename1) against the second (filename2) and reports any differences between them. Both filenames can be pathnames. For example:

```
FC B:\FOO\BAR\FILE1.TXT \BAR\FILE2.TXT
```

FC takes FILE1.TXT in the \FOO\BAR directory of disk drive B: and compares it with FILE2.TXT in the \BAR directory. Since no drive is specified for filename2, FC assumes that the \BAR directory is on the disk in the default drive.

### 8.4 FC SWITCHES

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

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

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

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

## \*\*\*Data left in F2\*\*\*

/# # stands for a number from 1 to 9. This switch specifies the number of lines required to match for the files to be considered as matching again after a difference has been found. If this switch is not specified, it defaults to 3. This switch is used only in source comparisons.

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

\_\_\_ More \_\_\_ data \_\_\_ to \_\_\_ be \_\_\_ found \_\_\_

will match with:

More\_data\_to\_be\_found

and with:

\_\_\_\_\_ More \_\_\_\_\_ data \_\_\_ to \_\_\_ be \_\_\_ found \_\_\_\_\_

but will not match with:

\_\_\_ Moredata\_to\_be\_found

This switch is used only in source comparisons.

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

Much MORE\_data\_IS\_NOT\_FOUND

will match:

much\_more\_data\_is\_not\_found

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

\_\_\_ DATA\_was\_found \_\_\_

will match:

data\_was\_found

This switch is used only in source comparisons.

**8.5 DIFFERENCE REPORTING**

The File Comparison Utility reports the differences between the two files you specify by displaying the first filename, followed by the lines that differ between the files, followed by the first line to match in both files. FC then displays the name of the second file followed by the lines that are different, followed by the first line that matches. The default for the number of lines to match between the files is 3. (If you want to change this default, specify the number of lines with the /# switch.) For example:

```

...
...
-----<filename1>
<difference>
<1st line to match file2 in file1>

-----<filename2>
<difference>
<1st line to match file1 in file2>

-----
...
...

```

FC will continue to list each difference.

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

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

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

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

**8.6 REDIRECTING FC OUTPUT TO A FILE**

The differences and matches between the two files you specify will be displayed on your screen unless you redirect the output to a file. This is accomplished in the same way as MS-DOS command redirection (refer to Chapter 4, "Learning About Commands").

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

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

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

**8.7 EXAMPLES**Example 1:

Assume these two ASCII files are on disk:

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

To compare the two files and display the differences on the terminal screen, type:

FC ALPHA.ASM BETA.ASM

FC compares ALPHA.ASM with BETA.ASM and displays the differences on the terminal screen. All other defaults remain intact. (The defaults are: do not compress whites and do not ignore case.)

The output will appear as follows on the terminal screen (the Notes do not appear):

-----ALPHA.ASM

D NOTE: ALPHA file  
E contains defg,  
F BETA contains g.  
G

-----BETA.ASM

G

-----ALPHA.ASM

M NOTE: ALPHA file  
N contains mno where  
O BETA contains jl2.  
P

-----BETA.ASM

J  
1  
2  
P

-----ALPHA.ASM

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

4  
5  
W

Example 2:

You can print the differences on the line printer using the same two source files. In this example, four successive lines must be the same to constitute a match.

Type:

```
FC /4 ALPHA.ASM BETA.ASM >PRN
```

The following output will appear on the line printer:

```
-----ALPHA.ASM
```

```
D
E
F
G
H
I
M
N
O
P
```

```
NOTE: p is the 1st of
a string of 4 matches.
```

```
-----BETA.ASM
```

```
G
H
I
J
l
2
P
```

```
-----ALPHA.ASM
```

```
W
```

```
NOTE: w is the 1st of a
string of 4 matches.
```

```
-----BETA.ASM
```

```
4
5
W
```

Example 3:

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

Type:

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

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

```
--ADDRS----F1---F2--  
00000009 44 47  
0000000C 45 48  
0000000F 46 49  
00000012 47 4A  
00000015 48 31  
00000018 49 32  
0000001B 4D 50  
0000001E 4E 51  
00000021 4F 52  
00000024 50 53  
00000027 51 54  
0000002A 52 55  
0000002D 53 56  
00000030 54 34  
00000033 55 35  
00000036 56 57  
00000039 57 58  
0000003C 58 59  
0000003F 59 5A  
00000042 5A 1A
```



**8.8 ERROR MESSAGES**

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

Bad file

One of the files you specified is defective.

Data left in (filename)

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

File not found:<filename>

FC could not find the filename you specified.

Incorrect DOS version

You are running FC under a version of MS-DOS that is not 2.0 or higher.

Internal error

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

Invalid number of parameters

You have specified the wrong number of options on the FC command line.

Invalid parameter:<option>

One of the switches that you have specified is invalid.

Read error in:<filename>

FC could not read the entire file.

## APPENDIX A

### INSTRUCTIONS FOR USERS WITH SINGLE-DRIVE SYSTEMS

#### NOTE

Information in this appendix is installation-dependent and may not be implemented on every manufacturer's machine.

On a single-drive system, you enter the commands as you would on a multi-drive system.

You should think of the single-drive system as having two drives (drive A: and drive B:). But instead of A: and B: representing two physical drives as on the multi-drive system, the A: and B: represent disks.

If you specify drive B: when the "drive A: disk" was last used, you are prompted to insert the disk for drive B:. For example:

```
A> COPY COMMAND.COM B:
Insert diskette for drive B:
and strike any key when ready
  1 File(s) copied
A>_
```

If you specify drive A: when the "drive B: disk" was last used, you are prompted again to change disks. This time, MS-DOS prompts you to insert the "drive A: disk."

The same procedure is used if a command is executed from a batch file. MS-DOS waits for you to insert the appropriate disk and press any key before it continues. You will be prompted to do this.

#### NOTE

The letter displayed in the system prompt represents the default drive where MS-DOS looks to find a file whose name is entered without a drive specifier. The letter in the system prompt does not represent the last disk used.

For example, assume that A: is the default drive. If the last operation performed was DIR B:, MS-DOS believes the "drive B: disk" is still in the drive. However, the system prompt is still A:, because A: is still the default drive. If you type DIR, MS-DOS prompts you for the "drive A: disk" because drive A: is the default drive, and you did not specify another drive in the DIR command.

## APPENDIX B

### DISK ERRORS

If a disk or device error occurs at any time during a command or program, MS-DOS returns an error message in the following format:

```
<yyy> error <I/O action> drive <x>  
Abort,Ignore,Retry:_
```

In this message,<yyy> may be one of the following:

```
Write protect error  
Bad unit error  
Not ready error  
Bad command error  
Data error  
Bad call format error  
Seek error  
Non-DOS disk error  
Sector not found error  
No paper error  
Write fault error  
Read fault error  
Disk error
```

The <I/O-action> may be either of the following:

```
READING  
WRITING
```

The drive <x> indicates the drive in which the error has occurred.

MS-DOS waits for you to enter one of the following responses:

- A Abort. Terminate the program requesting the disk read or write.
- I Ignore. Ignore the bad sector and pretend the

error did not occur.

R Retry. Repeat the operation. This response is to be used when the operator has corrected the error (such as with NOT READY or WRITE PROTECT errors).

Usually, you will want to attempt recovery by entering responses in this order:

- R (to try again)
- A (to terminate program and try a new disk)

One other error message might be related to faulty disk read or write:

FILE ALLOCATION TABLE BAD FOR DRIVE x

This message means that the copy in memory of one of the allocation tables has pointers to nonexistent blocks. Possibly the disk was incorrectly formatted or not formatted before use. If this error persists, the disk is currently unusable and must be formatted prior to use.

**APPENDIX C**  
**ANSI ESCAPE SEQUENCES**

**NOTE**

Information in this appendix is installation-dependent and may not be implemented on every manufacturer's machine.

An ANSI escape sequence is a series of characters (beginning with an escape character or keystroke) that you can use to define functions to MS-DOS. Specifically, you can reassign keys, change graphics functions, and affect cursor movement.

This appendix explains how the ANSI escape sequences are defined for MS-DOS version 2.0. Examples on how to use ANSI escape sequences are included at the end of this appendix.

**Notes:**

1. The default value is used when no explicit value or a value of zero is specified.
2. Pn represents "numeric parameter." This is a decimal number specified with ASCII digits.
3. Ps represents "selective parameter." This is any decimal number that is used to select a subfunction. Multiple subfunctions may be selected by separating the parameters with semicolons.

**C.1 CURSOR FUNCTIONS**

The following escape sequences affect the cursor position on the screen:

CUP - Cursor Postion

ESC [ P1 ; Pc H

HVP - Horizontal & Vertical Postion

ESC [ P1 ; Pc f

CUP and HVP move the cursor to the position specified by the parameters. The first parameter specifies the line number, and the second parameter specifies the column number. The default value is 1. When no parameters are specified, the cursor is moved to the home postion.

CUU - Cursor Up

ESC [ Pn A

This sequence moves the cursor up one line without changing columns. The value of Pn determines the number of lines moved. The default value for Pn is 1. The CUU sequence is ignored if the cursor is already on the top line.

CUD - Cursor Down

ESC [ Pn B

This sequence moves the cursor down one line without changing columns. The value of Pn determines the number of lines moved. The default value for Pn is 1. The CUD sequence is ignored if the cursor is already on the bottom line.

CUF - Cursor Forward

ESC [ Pn C

The CUF sequence moves the cursor forward one column without changing lines. The value of Pn determines the number of columns moved. The default value for Pn is 1. The CUF sequence is ignored if the cursor is already in the far right column.

**CUB - Cursor Backward**

ESC [ Pn D

This escape sequence moves the cursor back one column without changing lines. The value of Pn determines the number of columns moved. The default value for Pn is 1. The CUB sequence is ignored if the cursor is already in the far left column.

**DSR - Device Status Report**

ESC [ 6 n

The console driver will output a CPR sequence (see below) on receipt of the DSR escape sequence.

**CPR - Cursor Position Report (from console driver to system)**

ESC [ Pn ; Pn R

The CPR sequence reports current cursor position via standard input. The first parameter specifies the current line and the second parameter specifies the current column.

**SCP - Save Cursor Position**

ESC [ s

The current cursor position is saved. This cursor position can be restored with the RCP sequence (see below).

**RCP - Restore Cursor Position**

ESC [ u

This sequence restores the cursor position to the value it had when the console driver received the SCP sequence.

**C.2 ERASING**

The following escape sequences affect erase functions:



ED - Erase Display

ESC [ 2 J

The ED sequence erases the screen, and the cursor goes to the home position.

EL - Erase Line

ESC [ K

This sequence erases from the cursor to the end of the line (including the cursor position).

### C.3 MODES OF OPERATION

The following escape sequences affect screen graphics:

SGR - Set Graphics Rendition

ESC [ Ps ; ... ; Ps m

The SGR escape sequence invokes the graphic functions specified by the parameter(s) described below. The graphic functions remain until the next occurrence of an SGR escape sequence.

Parameter	Parameter Function	
0	All Attributes off	
1	Bold on	
4	Underscore on	(monochrome displays only)
5	Blink on	
7	Reverse Video on	
8	Concealed on	(ISO 6429 standard)
30	Black foreground	(ISO 6429 standard)
31	Red foreground	(ISO 6429 standard)
32	Green foreground	(ISO 6429 standard)
33	Yellow foreground	(ISO 6429 standard)
34	Blue foreground	(ISO 6429 standard)
35	Magenta foreground	(ISO 6429 standard)
36	Cyan foreground	(ISO 6429 standard)
37	White foreground	(ISO 6429 standard)
40	Black background	(ISO 6429 standard)
41	Red background	(ISO 6429 standard)
42	Green background	(ISO 6429 standard)
43	Yellow background	(ISO 6429 standard)

44	Blue background	(ISO 6429 standard)
45	Magenta background	(ISO 6429 standard)
46	Cyan background	(ISO 6429 standard)
47	White background	(ISO 6429 standard)

### SM - Set Mode

ESC [ = Ps h  
or ESC [ = h  
or ESC [ = 0 h  
or ESC [ ? 7 h

The SM escape sequence changes the screen width or type to one of the following parameters:

Parameter	Parameter Function
0	40 x 25 black and white
1	40 x 25 color
2	80 x 25 black and white
3	80 x 25 color
4	320 x 200 color
5	320 x 200 black and white
6	640 x 200 black and white
7	wrap at end of line

### RM - Reset Mode

ESC [ = Ps l  
or ESC [ = l  
or ESC [ = 0 l  
or ESC [ ? 7 l

Parameters for RM are the same as for SM (Set Mode), except that parameter 7 will reset the wrap at the end of line mode.

**C.4 KEYBOARD REASSIGNMENT**

Although not part of the ANSI 3.64-1979 or ISO 6429 standard, the following keyboard reassignments are compatible with these standards.

The control sequence is:

```

    ESC [ Pn ; Pn ; ... Pn p
or   ESC [ "string" ; p
or   ESC [ Pn ; "string" ; Pn ; Pn ; "string" ; Pn p
or   any other combination of strings and decimal numbers

```

The final code in the control sequence (p) is one reserved for private use by the ANSI 3.64-1979 standard.

The first ASCII code in the control sequence defines which code is being mapped. The remaining numbers define the sequence of ASCII codes generated when this key is intercepted. Note that there is one exception: if the first code in the sequence is zero (NUL), then the first and second code make up an extended ASCII redefinition.

Examples:

1. Reassign the Q and q key to the A and a key (and vice versa):

```

    ESC [ 6 5 ; 8 1 p           A becomes Q
    ESC [ 9 7 ; 1 1 3 p        a becomes q
    ESC [ 8 1 ; 6 5 p          Q becomes A
    ESC [ 1 1 3 ; 9 7 p        q becomes a

```

2. Reassign the F10 key to to a DIR command followed by a carriage return:

```

    ESC [ 0 ; 6 8 ; " d i r " ; 1 3 p

```

The 0;68 is the extended ASCII code for the F10 key; 13 decimal is a carriage return.

## APPENDIX D

### HOW TO CONFIGURE YOUR SYSTEM

In many cases, there are installation-specific settings for MS-DOS that need to be configured at system startup. An example of this is a standard device driver, such as an online printer.

The MS-DOS configuration file (CONFIG.SYS) allows you to configure your system with a minimum of effort. With this file, you can add device drivers to your system at startup. The configuration file is simply an ASCII file that has certain commands for MS-DOS startup (boot). The boot process is as follows:

1. The disk boot sector is read. This contains enough code to read MS-DOS code and the installation's BIOS (machine-dependent code).
2. The MS-DOS code and BIOS are read.
3. A variety of BIOS initializations are done.
4. A system initialization routine reads the configuration file (CONFIG.SYS), if it exists, to perform device installation and other user options. Its final task is to execute the command interpreter, which finishes the MS-DOS boot process.

#### D.1 CHANGING THE CONFIG.SYS FILE

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

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

**BUFFERS = <number>**

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

**FILES = <number>**

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

**DEVICE = <filename>**

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

**BREAK = <ON or OFF>**

If ON is specified (the default is OFF), a check for CONTROL-C as input will be made every time the system is called. ON improves the ability to abort programs over previous versions of the MS-DOS.

**SHELL = <filename>**

This begins execution of the shell (top-level command processor) from <filename>.

**COUNTRY = <number>**

This number is set by the equipment manufacturer to allow for international date, time, currency, and case conversion. Acceptable values are 1-99. This statement is only supported in versions of MS-DOS that are higher than 2.0.

A typical configuration file might look like this:

```
Buffers = 10
Files = 10
Device = \BIN\NETWORK.SYS
Break = ON
Shell = A:\BIN\COMMAND.COM A:\BIN /P
```

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

This configuration file also sets the MS-DOS command EXEC to the COMMAND.COM file located on disk A: in the \BIN directory. The A:\BIN tells COMMAND.COM where to look for itself when it needs to re-read from disk. The /P tells COMMAND.COM that it is the first program running on the system so that it can process the MS-DOS EXIT command.

## APPENDIX E

### MS-DOS MESSAGE DIRECTORY

- [EDLIN] Abort edit (Y/N)?  
This message is displayed when you choose the Q (Quit) command in EDLIN. The Quit command exits the editing session without saving any editing changes. Specify Y (for Yes) or N (for No).
- [MS-DOS] Abort, Retry, Ignore?  
If a disk or device error occurs at any time during a command or program, MS-DOS returns this message and asks you to abort the command or program, retry it, or ignore the error.
- [PRINT] All files cancelled by operator  
This message is displayed when you specify the /T switch with the PRINT command.
- [CHKDSK] All specified files are contiguous  
All files are allocated contiguously on the disk without fragmentation.
- [CHKDSK] Allocation error in file, size adjusted  
An invalid sector number was found in the FAT. The file was truncated at the end of the last valid sector.
- [COMMAND] Are you sure (Y,N)?  
MS-DOS displays this message if you try to delete \*.\* (all files in the current directory). Specify Y (for Yes ) or N (for No).
- [LINK] Attempt to access data outside of segment bounds, possibly bad object module  
There is probably a bad object file.
- [MS-DOS] Bad call format reading drive (x):  
Device error. See Appendix B.

- [MS-DOS] Bad call format writing drive (x:)  
Device error. See Appendix B.
- [MS-DOS] Bad command error reading drive (x:)  
Device error. See Appendix B.
- [MS-DOS] Bad command error writing drive (x:)  
Device error. See Appendix B.
- [COMMAND] Bad command or file name  
The command cannot find the file you asked it to run. You either mistyped the filename or the file does not exist on the disk.
- [FC] Bad file  
One of the files you specified is defective.
- [LINK] Bad numeric parameter  
Numeric value is not in digits.
- [MS-DOS] Bad or missing (filename)  
You specified an invalid device in the CONFIG.SYS file. Check the accuracy of the DEVICE statement in the CONFIG.SYS file.
- [MS-DOS] Bad or missing Command Interpreter  
MS-DOS cannot find the COMMAND.COM file on the disk; either the file is missing from the root directory, or the file is invalid. Either restart the system or copy the COMMAND.COM file from your backup MS-DOS system disk onto the disk used to start MS-DOS. You will also receive this message if COMMAND.COM has been moved from the directory it was originally in when you started MS-DOS.
- [MS-DOS] Bad unit error reading drive (x:)  
Device error. See Appendix B.
- [MSDOS] Bad unit error writing drive (x:)  
Device error. See Appendix B.
- [COMMAND] BREAK is off (or on)  
This message tells you the current setting of BREAK.



- [CHKDSK] Cannot CHDIR to (filename) - tree past this point not processed  
CHKDSK is traveling the tree structure of the directory and is unable to proceed to the specified directory. All subdirectories underneath this directory will not be verified.
- [CHKDSK] Cannot CHDIR to root  
Processing cannot continue  
CHKDSK is traveling the tree structure of the directory and is unable to return to the root directory. CHKDSK is not able to continue checking the remaining subdirectories to the root.
- [COMMAND] Cannot do binary reads from a device  
This message appears during COPY command processing. The COPY cannot be done in binary mode when you are copying from a device. Remove the /B switch or specify an ASCII copy with the /A switch.
- [EDLIN] Cannot edit .BAK file--rename file  
You attempted to edit a backup copy created by EDLIN. Either rename the file or copy the .BAK file and give it a different extension.
- [PRINT] Cannot open (filename)  
Either MS-DOS cannot find the specified file to print or the file does not exist. Check the command for a valid filename.
- [LINK] Cannot open temporary file  
MS-LINK is unable to create the file VM.TMP because the disk directory is full. Insert a new disk. Do not remove the disk that will receive the List.MAP file.
- [CHKDSK] Cannot recover . entry, processing continued  
The . entry (current directory) is defective.
- [CHKDSK] Cannot recover .. entry  
The .. entry (parent directory) is defective.
- [CHKDSK] CHDIR .. failed, trying alternate method  
In traveling the tree structure, CHKDSK was not able to return to a parent directory. It will try to return to that directory by starting over at the root and traveling down.

- [COMMAND] Content of destination lost before copy  
A file to be used as a source file to the COPY command has been overwritten prior to completion of the copy. Example:  
COPY A + B B  
which destroys B before it can be copied.
- [CHKDSK] Convert lost chains to files (Y/N)?  
If you respond Y to this prompt, CHKDSK will recover the lost blocks it found when checking the disk. CHKDSK will create a directory entry and a file for you with the filename FILEnnnn. If you respond N, CHKDSK frees the lost blocks so they can be reallocated.
- [DISKCOPY] Copy another (Y/N)?  
Respond Y if you wish to copy another disk. Respond N if you do not wish to copy another disk.
- [DISKCOPY] Copy complete  
DISKCOPY has completed processing.
- [DISKCOPY] Copy not completed  
DISKCOPY could not copy the entire disk.
- [DISKCOPY] Copying...  
This message indicates that DISKCOPY is copying a disk.
- [MS-DOS] Copyright 1981,82 Microsoft Corp.  
This message appears on most MS-DOS utility and command banners.
- [COMMAND] Current date is (mm-dd-yy)  
This message is displayed in response to the DATE command.
- [COMMAND] Current time is (hh:mm:ss.hh)  
This message is displayed in response to the TIME command.
- [MS-DOS] Data error reading drive (x):  
Device error. See Appendix B.
- [MS-DOS] Data error reading drive (x):  
Device error. See Appendix B.
- [FC] Data left in (filename)  
After reaching the end of one of the files in a file comparison, the other file still has uncompered data left.

- [CHKDSK] Directory is totally empty, no . or ..  
The specified directory does not contain references to current and parent directories. Delete the specified directory and recreate it.
- [MS-DOS] Disk error reading drive (x:)  
Device error. See Appendix B.
- [CHKDSK] Disk error reading FAT (x)  
One of your File Allocation Tables has a defective sector in it. MS-DOS will automatically use the other FAT. It is a good idea to copy all your files onto another disk.
- [MS-DOS] Disk error writing drive (x:)  
Device error. See Appendix B.
- [CHKDSK] Disk error writing FAT (x:)  
One of your File Allocation Tables has a defective sector in it. MS-DOS will automatically use the other FAT. It is a good idea to copy all your files onto another disk.
- [EDLIN] Disk full--write not completed  
You gave the End command, but the disk did not contain enough free space for the file. EDLIN aborted the E command and returned you to the operating system. Part of the file may have been written to disk and saved. Delete the saved portion and restart the editing session. The file will not be available after this error.
- [FORMAT] Disk unsuitable for system drive  
FORMAT detected a bad track on the disk where system files should reside. You should only store data on the disk you tried to format when this message was displayed.
- [DISKCOPY] Disks must be the same size  
You cannot copy the contents of a disk with a different format using DISKCOPY. Use the COPY command to copy files onto the disk.
- [MS-DOS] Divide overflow  
The 8086 has set the divide overflow flag which is usually caused by division by zero.
- [CHKDSK] (.) (..) Does not exist  
This is an informational message from CHKDSK. This message indicates either the . or .. directory entry is invalid.

- [COMMAND] Duplicate file name or File not found  
You have tried to rename a file to a filename that already exists or the name you specified could not be found.
- [COMMAND] ECHO is off (or on)  
This message tells you the current status of ECHO.
- [EDLIN] End of input file  
The entire file was read into memory. If the file is read in sections, this message indicates the last section of the file is in memory.
- [COMMAND] Enter new date:  
You must respond to this prompt when you start MS-DOS. Enter the date in a <mm>/<dd>/<yy> format.
- [COMMAND] Enter new time:  
You must respond to this prompt when you start MS-DOS. Enter the time in the <hh>:<mm>:<ss> format.
- [EDLIN] Entry error  
The last command you typed contained a syntax error. Retype the command with the correct syntax and press <RETURN>.
- [CHKDSK] Entry has a bad attribute (or link or size)  
This message may be preceded by one or two periods which indicate which subdirectory is invalid. If you have specified the /F switch, CHKDSK will attempt to correct the error.
- [LINK] Error: duplicate record too complex  
A DUP record in an assembly language module is too complex. Simplify the DUP record in your assembly language program.
- [LINK] Error: fixup offset exceeds field width  
An assembly language instruction refers to an address with a short instruction instead of a long instruction. Edit your assembly language source and reassemble.

- [COMMAND] Error in .EXE file  
The .EXE file you have asked MS-DOS to load has an invalid internal format.
- [PRINT] (type of error) error reading file  
Device error. See Appendix B.
- [COMMAND] Error writing to device  
You tried to send too much data to a device. MS-DOS was unable to write the data to the specified device.
- [CHKDSK] Errors found, F parameter not specified  
Corrections will not be written to disk  
CHKDSK found errors on the disk. If you have not specified the /F switch, CHKDSK will continue printing messages but will not correct the errors.
- [PRINT] Errors on list device indicate that it may be off-line. Please check it  
Your printer is offline.
- [COMMAND] EXEC failure  
MS-DOS either found an error when reading a command or the FILES statement in the CONFIG.SYS file is set too low. Increase the value and restart MS-DOS.
- [COMMAND] File allocation table bad  
The disk may be defective. Run CHKDSK to check the disk.
- [PRINT] File allocation table bad drive (x:) [CHKDSK]  
The disk may be defective. Run CHKDSK to check the disk.
- [COMMAND] File cannot be copied onto itself  
The source filename you specified is the same as the destination filename. Example:  
COPY A A.

- [EXE2BIN] File cannot be converted  
The input file is not in the correct format.
- [COMMAND] File creation error  
You tried to add a new filename or replace a file that already exists in the directory. If the file already exists, it is a read-only file and cannot be replaced. Run CHKDSK on the disk to determine the cause of the error.
- [CHKDSK] (filename) contains non-contiguous blocks  
The filename specified is not allocated contiguously on the disk. If you specify the /F switch, CHKDSK will fix this error.
- [PRINT] (filename) file not found  
You switched disks while a file was queued up, but before it started to print. Reissue the PRINT command for that filename.
- [CHKDSK] (filename) is cross linked on cluster  
Make a copy of the file you want to keep, and then delete both files that are cross linked.
- [PRINT] (filename) is currently being printed  
The filename specified is being printed.
- [PRINT] (filename) is in queue  
The filename specified is waiting to be printed.
- [EDLIN] File name must be specified  
You did not specify a filename when you started EDLIN.
- [COMMAND] File not found [EDLIN, FC, FIND, RECOVER]  
MS-DOS cannot find the file that you specified. Check to see that the pathname is accurate and that the file exists in the directory you specified.
- [FC] Files are different  
FC compares what can be loaded into the buffer space. If no lines match in the portion of the files in the buffer space, FC will display this message.

- [CHKDSK] First cluster number is invalid, entry truncated  
The file directory entry contains an invalid pointer to the data area. If you specified the /F switch, the file is truncated to a zero-length file.
- [EXE2BIN] 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.
- [COMMAND] FOR cannot be nested  
Nesting of FOR statements is not allowed in a batch file.
- [FORMAT] Format another (Y/N)?  
Type Y (for Yes) to format another disk. Type N (for No) if you do not want to format another disk. If you accidentally type Y, you can abort the format process by typing <CONTROL-C> in response to the "Strike any key to begin formatting" message.
- [FORMAT] Format failure  
MS-DOS could not format the disk. This message is always displayed in conjunction with an explanation as to why MS-DOS could not format the disk.
- [SYS] Incompatible system size  
The system files IO.SYS and MSDOS.SYS occupy more space on the source disk than is available on the destination disk.
- [CHKDSK] Incorrect DOS version [EDLIN, FC, FIND, FORMAT, MORE, PRINT, RECOVER, SORT, SYS]  
Many version 2.00 utilities will not run on older versions of MS-DOS. The utilities CHKDSK, PRINT, and SYS will only run under the exact version of MS-DOS for which they were configured.
- [LINK] Input file read error  
There is probably a defective object file.

- [MS-DOS] Insert diskette for drive (x:) and strike any key when ready  
This message appears when MS-DOS is copying and formatting. You should insert a disk in the appropriate drive and press any alphanumeric key to begin processing.
- [COMMAND] Insert diskette with batch file and press any key when ready  
You no longer have the disk containing the batch file you specified in the drive you originally specified. Reinsert the disk that contains the batch file in the appropriate drive.
- [FORMAT] Insert DOS diskette in drive (x:) and strike any key when ready  
You have specified FORMAT /S but the disk in the default drive does not contain MS-DOS system files. Insert a disk with the files IO.SYS and MSDOS.SYS (bootable disk) in the drive specified.
- [DISKCOPY] Insert formatted target diskette into drive (x:)  
DISKCOPY is ready for a disk in the destination drive. DISKCOPY requires that the destination disk be already formatted.
- [FORMAT] Insert new diskette for drive (x:) and strike any key when ready  
Insert a blank disk into the appropriate drive and press any alphanumeric key to begin formatting. If there is any data on the disk, it will be destroyed by the format process.
- [DISKCOPY] Insert source diskette into drive (x:)  
Insert the disk to be copied into the specified drive.
- [SYS] Insert system diskette in drive (x:) and strike any key when ready  
SYS needs a bootable disk from which to read the IO.SYS and MSDOS.SYS files. Insert a bootable disk into the specified drive and press any alphanumeric key to start the system copy process.
- [DISKCOPY] Insert target diskette into drive (x:)  
You are running DISKCOPY and your source and destination drives are the same. Reinsert the destination disk into the specified drive.



## INDEX

Abort . . . . . B-1  
 Adding lines, (EDLIN) . . . . 7-22  
 ANSI escape sequences . . . . C-1  
   cursor functions . . . . . C-2  
   erasing . . . . . C-3  
   modes of operation . . . . . C-4  
 Arguments . . . . . 4-4  
 Asterisk (\*) . . . . . 3-3 to 3-4  
 AUTOEXEC.BAT file . . . . . 2-8, 4-8, 4-11, 5-20  
 Automatic Program Execution . . 2-8, 4-8  
  
 BACKUP command . . . . . 5-5  
 Backup disks . . . . . 2-6, 3-7  
 BASIC . . . . . 4-11  
 .BAT extension . . . . . 4-6  
 .BAT file  
   executing . . . . . 4-13  
 Batch processing . . . . . 4-6  
 Batch processing commands . . . 5-58  
 Binary files (FC) . . . . . 8-2  
 BIOS initializations . . . . . D-1  
 Brackets . . . . . 3-3  
 BREAK (CONFIG.SYS file) . . . . D-2  
 BREAK command . . . . . 5-7  
 Buffer space (FC) . . . . . 8-2  
 BUFFERS (CONFIG.SYS file) . . . D-2  
  
 Changing directories . . . . . 3-14  
 CHDIR command . . . . . 3-13 to 3-14, 5-8  
 Checking disks . . . . . 2-11  
 CHKDSK command . . . . . 2-11  
 CHKDSK errors . . . . . 5-9  
 Clearing the screen . . . . . 5-14  
 CLS command . . . . . 5-14  
 Command formats . . . . . 5-2  
 Command options . . . . . 4-3  
 Command options (EDLIN) . . . . 7-20  
 Command processor . . . . . 2-2, 2-4, 3-12, 4-2

COMMAND.COM	2-2
Commands	4-2
BACKUP	5-5
BREAK	5-7
CD	3-14
CHDIR	5-8
CHKDSK	2-11, 5-9
CLS	5-14
COPY	3-5, 5-15
CTTY	5-19
DATE	5-20
DEL	5-22
DIR	2-9, 3-13, 5-23
DISKCOPY	2-6, 5-24
ECHO	5-58
EXE2BIN	5-27
EXIT	5-30
External	4-2
FIND	4-14, 5-31
FOR	5-59
FORMAT	2-4, 2-9, 5-33
GOTO	5-60
IF	5-61
Internal	4-2
MKDIR	3-13, 5-35
MORE	4-14, 5-36
PATH	5-37
PAUSE	4-6, 5-62
PRINT	5-38
PROMPT	5-41
RECOVER	5-43
REM	4-6, 5-44
REN	5-45
RESTORE	5-46
RMDIR	3-14, 5-47
SET	5-48
SHIFT	4-12, 5-63
SORT	4-14, 5-49
SYS	5-51
TYPE	5-54
VER	5-55
VERIFY	5-56
VOL	5-57
Comparing files	8-2
Concatenation	5-16

CONFIG.SYS file	
changing . . . . .	D-1
Configuration file . . . . .	D-2
Configuring your system . . . . .	D-1
Control character functions . . . . .	6-6 to 6-7
<CONTROL-C> . . . . .	4-5, 4-8
CONTROL-C check . . . . .	5-7
<CONTROL-H> . . . . .	6-7
<CONTROL-J> . . . . .	6-7
<CONTROL-N> . . . . .	6-7
<CONTROL-P> . . . . .	6-7
<CONTROL-S> . . . . .	4-5, 6-7
<CONTROL-X> . . . . .	6-7
<CONTROL-Z> . . . . .	4-11 to 4-12, 6-4
COPY command . . . . .	3-5
<COPY1> . . . . .	6-4, 7-5 to 7-6
<COPYALL> . . . . .	6-4, 7-5, 7-8
Copying disks . . . . .	2-6, 5-24
Copying files . . . . .	5-15
Copying lines, (EDLIN) . . . . .	7-23
<COPYUP> . . . . .	6-4, 7-5, 7-7
CTTY command . . . . .	5-19
Current date . . . . .	2-2
Current time . . . . .	2-3
Cursor Backward sequence . . . . .	C-3
Cursor Down sequence . . . . .	C-2
Cursor Forward sequence . . . . .	C-2
Cursor functions . . . . .	C-2
Cursor movement . . . . .	C-1
Cursor Position Report sequence . . . . .	C-3
Cursor Up sequence . . . . .	C-2
Data Error . . . . .	B-1
Data protection . . . . .	2-6
Date . . . . .	2-2
Default drive . . . . .	2-4, 8-3
Deleting directories . . . . .	5-47
Deleting files . . . . .	5-22
Deleting lines (EDLIN) . . . . .	7-25
Delimiters . . . . .	4-4
DEVICE (CONFIG.SYS file) . . . . .	D-2
Device errors . . . . .	B-1
Device names . . . . .	3-5
Device Status Report sequence . . . . .	C-3
Difference reporting (FC) . . . . .	8-5
DIR command . . . . .	2-9, 3-13

<b>Directories</b>	
changing . . . . .	3-14, 5-8
creating . . . . .	3-13
definition . . . . .	2-9, 3-7
displaying . . . . .	2-9, 5-23
hierarchical . . . . .	3-7, 3-10
making . . . . .	5-35
removing . . . . .	3-14, 5-47
root . . . . .	2-9
working . . . . .	2-9, 3-8, 3-13
<b>Disk Errors</b>	
Abort . . . . .	B-1
Data Error . . . . .	B-1
Disk Error . . . . .	B-1
File Allocation Table Bad For Drive x	B-2
Ignore . . . . .	B-1
Not Ready Error . . . . .	B-1
Retry . . . . .	B-2
Sector Not Found Error . . . . .	B-1
Seek Error . . . . .	B-1
Write Fault Error . . . . .	B-1
Write Protect Error . . . . .	B-1
DISKCOPY command . . . . .	2-6, 5-24
<b>Disks</b>	
backing up . . . . .	2-6, 5-5
checking . . . . .	2-11, 5-9
copying . . . . .	2-6, 5-24
formatting . . . . .	2-4, 5-33
Displaying directories . . . . .	5-23
Displaying files . . . . .	5-54
Drive designation . . . . .	3-2, 4-3
Drive designation (FC) . . . . .	8-3
Drive designation (MS-LINK) . . . . .	9-7
Dummy parameters . . . . .	4-12
ECHO command . . . . .	5-58
Editing and function keys . . . . .	4-5
<b>Editing commands</b>	
Copy characters . . . . .	7-6
Copy template . . . . .	7-8
Enter insert mode . . . . .	7-12
Exit insert mode . . . . .	7-14
New template . . . . .	7-15
Quit . . . . .	7-11
Quit input . . . . .	7-11
Replace mode . . . . .	7-14
Skip multiple characters . . . . .	7-10
Skip one character . . . . .	7-9
Editing files . . . . .	7-2
Editing keys . . . . .	6-2, 7-3
Editing text (EDLIN) . . . . .	7-27
EDLIN . . . . .	4-6, 7-2
starting . . . . .	7-2

EDLIN commands	
Append Lines . . . . .	7-22
Copy Lines . . . . .	7-23
Delete Lines . . . . .	7-25
Edit Line . . . . .	7-27
End Editing . . . . .	7-29
Insert Text . . . . .	7-30
List Text . . . . .	7-33
Page . . . . .	7-37
Quit . . . . .	7-38
Replace Text . . . . .	7-39
Search Text . . . . .	7-42
Transfer Text . . . . .	7-45
Write Lines . . . . .	7-46
Ellipsis . . . . .	5-2
Ending the edit session, (EDLIN)	7-29
Erase Display sequence . . . . .	C-4
Erase Line sequence . . . . .	C-4
Erasing the screen . . . . .	C-3
Error messages (FC) . . . . .	8-10
Error messages (EDLIN) . . . . .	7-47
Error messages, MS-DOS . . . . .	E-1
EXE2BIN command . . . . .	5-27
EXE2BIN errors . . . . .	5-28
Executable files	
converting . . . . .	5-27
definition . . . . .	4-3
Executing a .BAT file . . . . .	4-13
EXIT command . . . . .	5-30
External commands . . . . .	3-11, 4-2 to 4-3
FC . . . . .	8-2
FC switches . . . . .	8-3
File Allocation Table . . . . .	2-9
File Allocation Table Bad For Drive x	B-2
File specification . . . . .	3-3, 8-2
File system . . . . .	3-8
Filename extension . . . . .	4-3
Filename extensions	
.COM . . . . .	4-3
.EXE . . . . .	4-3

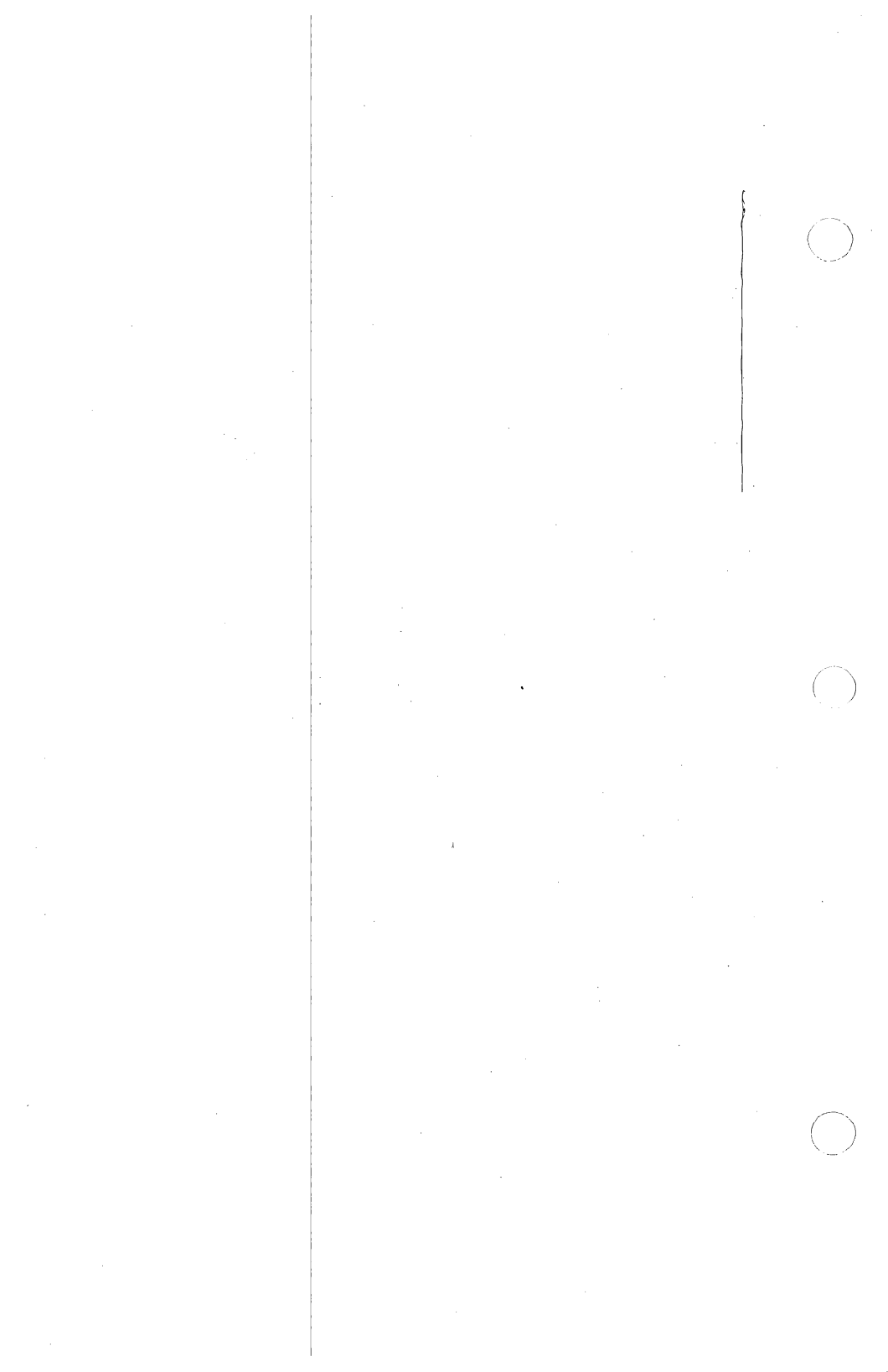
<b>Filenames</b>	
as part of commands . . . . .	4-3
definition . . . . .	3-2
extension . . . . .	3-2
illegal . . . . .	3-5
parts . . . . .	3-2
<b>Files</b>	
binary (FC) . . . . .	8-2
comparing . . . . .	8-2
concatenating . . . . .	5-16
copying . . . . .	3-5, 5-15
definition . . . . .	2-9
deleting . . . . .	5-22
displaying . . . . .	5-54
editing . . . . .	7-2
naming . . . . .	3-2
printing . . . . .	5-38
protecting . . . . .	3-7
recovering . . . . .	5-43
renaming . . . . .	5-45
restoring . . . . .	5-46
sorting . . . . .	5-49
source (FC) . . . . .	8-2
<b>FILES (CONFIG.SYS file)</b> . . . . .	D-2
<b>Filespec</b> . . . . .	4-4
<b>Filters</b> . . . . .	4-14
definition . . . . .	4-14
<b>FIND</b> . . . . .	4-14
<b>MORE</b> . . . . .	4-14
<b>SORT</b> . . . . .	4-14
<b>FIND command</b> . . . . .	4-14, 5-31
<b>FIND errors</b> . . . . .	5-32
<b>FOR command</b> . . . . .	5-59
<b>FORMAT command</b> . . . . .	2-4, 2-9
<b>Formatting disks</b> . . . . .	2-4, 5-33
<b>GOTO command</b> . . . . .	5-60
<b>Graphic functions, changing</b> . . . . .	C-1
<b>Hidden files</b> . . . . .	2-10
<b>Hierarchical directory</b> . . . . .	3-7, 3-9
<b>IF command</b> . . . . .	5-61
<b>Ignore</b> . . . . .	B-1
<b>Illegal filenames</b> . . . . .	3-5
<b>Input</b> . . . . .	4-13
<b>Input redirection</b> . . . . .	4-14
<b>&lt;INSERT&gt;</b> . . . . .	6-4, 7-5, 7-12
<b>Inserting text (EDLIN)</b> . . . . .	7-30
<b>Internal commands</b> . . . . .	3-12, 4-2

Joining files . . . . .	5-16
Keyboard reassignment . . . . .	C-6
Keys, reassigning . . . . .	C-1
Keywords . . . . .	5-2
LINK (FC) . . . . .	8-2
Listing lines (EDLIN) . . . . .	7-33
Loading MS-DOS . . . . .	2-2
Logoff . . . . .	2-11
Making directories . . . . .	5-35
Merging files (EDLIN) . . . . .	7-45
Message directory . . . . .	E-1
MKDIR command . . . . .	3-13, 5-35
Modes of operation . . . . .	C-4
MORE command . . . . .	4-14, 5-36
Moving lines, (EDLIN) . . . . .	7-36
MS-DOS commands . . . . .	5-2
MS-DOS files . . . . .	1-4
MS-DOS prompt . . . . .	2-4
Multiplan . . . . .	4-8
<NEWLINE> . . . . .	6-4, 7-5, 7-15
Not Ready Error . . . . .	B-1
/O (FORMAT) . . . . .	5-33
Operating system . . . . .	1-2
Output . . . . .	4-13
Output redirection . . . . .	4-14
Output redirection (FC) . . . . .	8-6
Paging through files (EDLIN) . . . . .	7-37
Parameters	
definition . . . . .	4-12
replaceable . . . . .	4-12
Parent directory	
CHDIR command . . . . .	5-8
shorthand notation . . . . .	3-11
PATH command . . . . .	3-11, 5-37
Pathing . . . . .	3-10
Pathnames	
definition . . . . .	3-10
syntax . . . . .	3-10, 4-4
used with CHDIR command . . . . .	3-14
PAUSE command . . . . .	4-6
Piping . . . . .	4-13, 4-15
PRINT errors . . . . .	5-39
PRINT command . . . . .	5-38

PROMPT command . . . . .	5-41
Protecting files . . . . .	3-7
Question mark (?) . . . . .	3-3
Quitting the edit session (EDLIN)	7-38
RECOVER command . . . . .	5-43
Recovering files . . . . .	5-43
Redirecting output . . . . .	4-14
Redirecting output (FC) . . . . .	8-6
Relative address (FC) . . . . .	8-3
REM command . . . . .	4-6
Removing directories . . . . .	3-14, 5-47
RENAME (synonym for REN) . . . . .	5-45
Renaming files . . . . .	5-45
<REPLACE> . . . . .	7-5, 7-14
Replaceable parameters . . . . .	4-12
Replacing text (EDLIN) . . . . .	7-39
Reserved filenames	
AUX . . . . .	3-5
CON . . . . .	3-5
NUL . . . . .	3-5
PRN . . . . .	3-5
Reset Mode sequence . . . . .	C-5
RESTORE command . . . . .	5-46
Restore Cursor Position sequence	C-3
Restoring files . . . . .	5-46
Retry . . . . .	B-2
RMDIR command . . . . .	3-14, 5-47
Root directory . . . . .	2-9, 3-9, D-1
/S (FORMAT) . . . . .	5-33
Save Cursor Position sequence	C-3
Screen	
changing . . . . .	C-1
erasing . . . . .	C-3
graphics . . . . .	C-4
Searching for text (EDLIN) . . . . .	7-42
Sector Not Found Error . . . . .	B-1
Seek Error . . . . .	B-1
Separators . . . . .	2-2
SET command . . . . .	5-48
Set Graphics Rendition sequence	C-4
Set Mode sequence . . . . .	C-5
SHELL (CONFIG.SYS file) . . . . .	D-2
SHIFT command . . . . .	4-12, 5-63
Shorthand notation . . . . .	3-11
Single-drive systems . . . . .	A-1



<SKIPL>	6-4, 7-5, 7-9
<SKIPUP>	6-4, 7-5, 7-10
SORT command	4-14, 5-49
Sorting files	5-49
Source drives	4-5
Source files (FC)	8-2
Special characters	3-2
Special editing keys	7-3
Starting	
EDLIN	7-2
FC	8-3
MS-DOS	2-2
Subdirectory	3-13
Switches	4-4
FC	
/#	8-4
/B	8-3
/C	8-4
/W	8-4
Syntax notation	1-4
Syntax of pathnames	3-10
SYS command	5-51
SYS errors	5-51
Target drives	4-5
Time	2-2
Transferring system files	5-51
/V (FORMAT)	5-33
VER command	5-55
VERIFY command	5-56
Version number, displaying	5-55
<VOID>	6-4, 7-5, 7-11
VOL command	5-57
Volume ID	3-13
Volume label	
definition	2-5
displaying	5-57
Wild cards	3-3, 4-5
the * wild card	3-4
the ? wild card	3-3
Working directory	
creating files	3-8
definition	2-9
displaying	3-13, 5-8
shorthand notation	3-11
Write Fault Error	B-1
Write Protect Error	B-1
Writing to disk (EDLIN)	7-46



**Section 3**

**Microsoft DEBUG  
(plus PCW-1 supplement)**

# Microsoft® DEBUG

---

Utility

Microsoft Corporation

## System Requirements

The MS-DOS for the PCW-1 disk is designed specifically for the Minolta Office System PCW-1.

# Contents

## Chapter 1

### INTRODUCTION

1.1	Overview of DEBUG	1-1	
1.2	How to Start DEBUG	1-1	
1.2.1	Method 1: DEBUG	1-2	
1.2.2	Method 2: Command Line	1-2	

## Chapter 2

### COMMANDS

2.1	Command Information	2-1	
2.2	Parameters	2-3	
2.3	Notes for PCW-1 Users	2-37	
2.4	Error Messages	2-38	

## CHAPTER 1

### INTRODUCTION

#### 1.1 OVERVIEW OF DEBUG

The Microsoft DEBUG Utility (DEBUG) is a debugging program that provides a controlled testing environment for binary and executable object files. Note that EDLIN is used to alter source files; DEBUG is EDLIN's counterpart for binary files. DEBUG eliminates the need to reassemble a program to see if a problem has been fixed by a minor change. It allows you to alter the contents of a file or the contents of a CPU register, and then to immediately reexecute a program to check on the validity of the changes.

All DEBUG commands may be aborted at any time by pressing <CONTROL-C>. <CONTROL-S> suspends the display, so that you can read it before the output scrolls away. Entering any key other than <CONTROL-C> or <CONTROL-S> restarts the display. All of these commands are consistent with the control character functions available at the MS-DOS command level.

#### 1.2 HOW TO START DEBUG

DEBUG may be started two ways. By the first method, you type all commands in response to the DEBUG prompt (a hyphen). By the second method, you type all commands on the line used to start DEBUG.

Summary of Methods to Start DEBUG

```
=====
Method 1          DEBUG
Method 2          DEBUG [<filespec> [<arglist>]]
=====
```

### 1.2.1 Method 1: DEBUG

To start DEBUG using method 1, type:

```
DEBUG
```

DEBUG responds with the hyphen (-) prompt, signaling that it is ready to accept your commands. Since no filename has been specified, current memory, disk sectors, or disk files can be worked on by using other commands.

#### Warnings

1. When DEBUG (Version 2.0) is started, it sets up a program header at offset 0 in the program work area. On previous versions of DEBUG, you could overwrite this header. You can still overwrite the default header if no <filespec> is given to DEBUG. If you are debugging a .COM or .EXE file, however, do not tamper with the program header below address 5CH, or DEBUG will terminate.
2. Do not restart a program after the "Program terminated normally" message is displayed. You must reload the program with the N and L commands for it to run properly.

### 1.2.2 Method 2: Command Line

To start DEBUG using a command line, type:

```
DEBUG [<filespec> [<arglist>]
```

For example, if a <filespec> is specified, then the following is a typical command to start DEBUG:

```
DEBUG FILE.EXE
```

DEBUG then loads FILE.EXE into memory starting at 100 hexadecimal in the lowest available segment. The BX:CX registers are loaded with the number of bytes placed into memory.

An <arglist> may be specified if <filespec> is present. The <arglist> is a list of filename parameters and switches that are to be passed to the program <filespec>. Thus, when <filespec> is loaded into memory, it is loaded as if it had been started with the command:



<filespec> <arglist>

Here, <filespec> is the file to be debugged, and the <arglist> is the rest of the command line that is used when <filespec> is invoked and loaded into memory.

## CHAPTER 2

### COMMANDS

#### 2.1 COMMAND INFORMATION

Each DEBUG command consists of a single letter followed by one or more parameters. Additionally, the control characters and the special editing functions described in the MS-DOS User's Guide, apply inside DEBUG.

If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with an up-arrow (^) and the word "error."

For example:

```
dcx:l00 cs:l10
  ^ error
```

Any combination of uppercase and lowercase letters may be used in commands and parameters.

The DEBUG commands are summarized in Table 2.1 and are described in detail, with examples, following the description of command parameters.

Table 2.1 DEBUG Commands

DEBUG Command	Function
A[<address>]	Assemble
C<range> <address>	Compare
D[<range>]	Dump
E<address> [<list>]	Enter
F<range> <list>	Fill
G[=<address> [<address>...]]	Go
H<value> <value>	Hex
I<value>	Input
L[<address> [<drive><record><record>]]	Load
M<range> <address>	Move
N<filename>[<filename>]	Name
O<value> <byte>	Output
P[=<address>][<value>]	PTrace
Q	Quit
R[<register-name>]	Register
S<range> <list>	Search
T[=<address>][ <value>]	Trace
U[<range>]	Unassemble
W[<address> [<drive><record><record>]]	Write

## 2.2 PARAMETERS

All DEBUG commands accept parameters, except the Quit command. Parameters may be separated by delimiters (spaces or commas), but a delimiter is required only between two consecutive hexadecimal values. Thus, the following commands are equivalent:

```

dcs:100 110
d cs:100 110
d,cs:100,110

```

PARAMETER	DEFINITION
<drive>	A one-digit hexadecimal value to indicate which drive a file will be loaded from or written to. The valid values are 0-3. These values designate the drives as follows: 0=A:, 1=B:, 2=C:, 3=D:.
<byte>	A two-digit hexadecimal value to be placed in or read from an address or register.
<record>	A 1- to 3-digit hexadecimal value used to indicate the logical record number on the disk and the number of disk sectors to be written or loaded. Logical records correspond to sectors. However, their numbering differs since they represent the entire disk space.
<value>	A hexadecimal value up to four digits used to specify a port number or the number of times a command should repeat its functions.
<address>	A two-part designation consisting of either an alphabetic segment register designation or a four-digit segment address plus an offset value. The segment designation or segment address may be omitted, in which case the default segment is used. DS is the default segment for all commands except G, L, T, U, and W, for which the default segment is CS. All numeric values are hexadecimal.

For example:

```

CS:0100
04BA:0100

```

The colon is required between a segment designation (whether numeric or alphabetic) and an offset.

<range> Two <address>es: e.g., <address> <address>; or one <address>, an L, and a <value>: e.g., <address> L <value> where <value> is the number of lines the command should operate on, and L80 is assumed. The last form cannot be used if another hex value follows the <range>, since the hex value would be interpreted as the second <address> of the <range>.

Examples:

```
CS:100 110
CS:100 L 10
CS:100
```

The following is illegal:

```
CS:100 CS:110
      ^ error
```

The limit for <range> is 10000 hex. To specify a <value> of 10000 hex within four digits, type 0000 (or 0).

<list> A series of <byte> values or of <string>s. <list> must be the last parameter on the command line.

Example:

```
fcs:100 42 45 52 54 41
```

<string> Any number of characters enclosed in quote marks. Quote marks may be either single (') or double("). If the delimiter quote marks must appear within a <string>, the quote marks must be doubled. For example, the following strings are legal:

```
'This is a "string" is okay.'
'This is a "'string'" is okay.'
```

However, this string is illegal:

```
'This is a 'string' is not.'
```

Similarly, these strings are legal:

```
"This is a 'string' is okay."
"This is a ""string"" is okay."
```

However, this string is illegal:

"This is a "string" is not."

Note that the double quote marks are not necessary in the following strings:

"This is a 'string' is not necessary."

'This is a "string" is not necessary.'

The ASCII values of the characters in the string are used as a <list> of byte values.

## NAME

Assemble

## PURPOSE

Assembles 8086/8087/8088 mnemonics directly into memory.  
(See note on page 2-37.)

## SYNTAX

A[&lt;address&gt;]

## COMMENTS

If a syntax error is found, DEBUG responds with

```
^Error
```

and redisplay the current assembly address.

All numeric values are hexadecimal and must be entered as 1-4 characters. Prefix mnemonics must be specified in front of the opcode to which they refer. They may also be entered 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 MOVSW to move word strings and MOVSB to move byte strings.

The assembler will automatically assemble short, near or far jumps and calls, depending on byte displacement to the destination address. These may be overridden with the NEAR or FAR prefix. For example:

```
0100:0500 JMP 502 ; a 2-byte short jump
0100:0502 JMP NEAR 505 ; a 3-byte near jump
0100:505 JMP FAR 50A ; a 5-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 also cannot tell whether an operand refers to a memory location or to an immediate operand. DEBUG uses the common convention that operands enclosed in square brackets refer to memory. For example:

```
MOV    AX,21           ; Load AX with 21H
MOV    AX,[21]        ; Load AX with the
                      ; contents
                      ; of memory location 21H
```

Two popular pseudo-instructions are available with Assemble. The DB opcode will assemble byte values directly into memory. The DW opcode will assemble word values directly into memory. For example:

```
DB    1,2,3,4,"THIS IS AN EXAMPLE"
DB    'THIS IS A QUOTE: "'
DB    "THIS IS A QUOTE: '"
DW    1000,2000,3000,"BACH"
```

Assemble supports all forms of register indirect commands. For example:

```
ADD    BX,34[BP+2].[SI-1]
POP    [BP+DI]
PUSH   [SI]
```

All opcode synonyms are also supported. For example:

```
LOOPZ  100
LOOPE  100

JA     200
JNBE   200
```

For 8087 opcodes, the WAIT or FWAIT must be explicitly specified. For example:

```
FWAIT FADD ST,ST(3) ; This line will assemble
                    ; an FWAIT prefix
LD TBYTE PTR [BX]  ; This line will not
```



## NAME

Compare

## PURPOSE

Compares the portion of memory specified by <range> to a portion of the same size beginning at <address>.

## SYNTAX

C&lt;range&gt; &lt;address&gt;

## COMMENTS

If the two areas of memory are identical, there is no display and DEBUG returns with the MS-DOS prompt. If there are differences, they are displayed in this format:

<address1> <byte1> <byte2> <address2>

## EXAMPLE

The following commands have the same effect:

C100,1FF 300

or

C100L100 300

Each command compares the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH.

## NAME

Dump

## PURPOSE

Displays the contents of the specified region of memory.

## SYNTAX

D[&lt;range&gt;]

## COMMENTS

If a range of addresses is specified, the contents of the range are displayed. If the D command is typed without parameters, 128 bytes are displayed at the first address (DS:100) after the address displayed by the previous Dump command.

The dump is displayed in two portions: a hexadecimal dump (each byte is shown in hexadecimal value) and an ASCII dump (the bytes are shown in ASCII characters). Nonprinting characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows 16 bytes with a hyphen between the eighth and ninth bytes. At times, displays are split in this manual to fit them on the page. Each displayed line begins on a 16-byte boundary.

If you type the command:

```
dcS:100 110
```

DEBUG displays the dump in the following format:

```
04BA:0100 42 45 52 54 41 ... 4E 44 TOM SAWYER
```

If you type the following command:

```
D
```

the display is formatted as described above. Each line of the display begins with an address, incremented by 16 from the address on the previous line. Each subsequent D (typed without parameters) displays the bytes immediately following those last displayed.

If you type the command:

```
DCS:100 L 20
```

the display is formatted as described above,  
but 20H bytes are displayed.

If then you type the command:

```
DCS:100 115
```

the display is formatted as described above,  
but all the bytes in the range of lines from  
100H to 115H in the CS segment are displayed.

## NAME

Enter

## PURPOSE

Enters byte values into memory at the specified <address>.

## SYNTAX

E&lt;address&gt;[ &lt;list&gt;]

## COMMENTS

If the optional <list> of values is typed, the replacement of byte values occurs automatically. (If an error occurs, no byte values are changed.)

If the <address> is typed without the optional <list>, DEBUG displays the address and its contents, then repeats the address on the next line and waits for your input. At this point, the Enter command waits for you to perform one of the following actions:

1. Replace a byte value with a value you type. Simply type the value after the current value. If the value typed in is not a legal hexadecimal value or if more than two digits are typed, the illegal or extra character is not echoed.
2. Press the <SPACE> bar to advance to the next byte. To change the value, simply type the new value as described in (1.) above. If you space beyond an 8-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.
3. Type a hyphen (-) to return to the preceding byte. If you decide to change a byte behind the current position, typing the hyphen returns the current position to the previous byte. When the hyphen is typed, a new line is started with the address and its byte value displayed.
4. Press the <RETURN> key to terminate the Enter command. The <RETURN> key may be pressed at any byte position.

## EXAMPLE

Assume that the following command is typed:

```
ECS:100
```

DEBUG displays:

```
04BA:0100 EB._
```

To change this value to 41, type 41 as shown:

```
04BA:0100 EB.41_
```

To step through the subsequent bytes, press the <SPACE> bar to see:

```
04BA:0100 EB.41 10. 00. BC._
```

To change BC to 42:

```
04BA:0100 EB.41 10. 00. BC.42_
```

Now, realizing that 10 should be 6F, type the hyphen as many times as needed to return to byte 0101 (value 10), then replace 10 with 6F:

```
04BA:0100 EB.41 10. 00. BC.42-
```

```
04BA:0102 00.-
```

```
04BA:0101 10.6F_
```

Pressing the <RETURN> key ends the Enter command and returns to the DEBUG command level.

## NAME

Fill

## PURPOSE

Fills the addresses in the <range> with the values in the <list>.

## SYNTAX

F&lt;range&gt; &lt;list&gt;

## COMMENTS

If the <range> contains more bytes than the number of values in the <list>, the <list> will be used repeatedly until all bytes in the <range> are filled. If the <list> contains more values than the number of bytes in the <range>, the extra values in the <list> will be ignored. If any of the memory in the <range> is not valid (bad or nonexistent), the error will occur in all succeeding locations.

## EXAMPLE

Assume that the following command is typed:

```
F04BA:100 L 100 42 45 52 54 41
```

DEBUG fills memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values are repeated until all 100H bytes are filled.

## NAME

Go

## PURPOSE

Executes the program currently in memory.

## SYNTAX

G[=&lt;address&gt;[ &lt;address&gt;...]]

## COMMENTS

If only the Go command is typed, the program executes as if the program had run outside DEBUG.

If =<address> is set, execution begins at the address specified. The equal sign (=) is required, so that DEBUG can distinguish the start =<address> from the breakpoint <address>es.

With the other optional addresses set, execution stops at the first <address> encountered, regardless of that address' position in the list of addresses to halt execution or program branching. When program execution reaches a breakpoint, the registers, flags, and decoded instruction are displayed for the last instruction executed. (The result is the same as if you had typed the Register command for the breakpoint address.)

Up to ten breakpoints may be set. Breakpoints may be set only at addresses containing the first byte of an 8086 opcode. If more than ten breakpoints are set, DEBUG returns the BP Error message.

The user stack pointer must be valid and have 6 bytes available for this command. The G command uses an IRET instruction to cause a jump to the program under test. The user stack pointer is set, and the user flags, Code Segment register, and Instruction Pointer are pushed on the user stack. (Thus, if the user stack is not valid or is too small, the operating system may crash.) An interrupt code (0CCH) is placed at the specified breakpoint address(es).

When an instruction with the breakpoint code is encountered, all breakpoint addresses are restored to their original instructions. If

execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

EXAMPLE

Assume that the following command is typed:

```
GCS:7550
```

The program currently in memory executes up to the address 7550 in the CS segment. DEBUG then displays registers and flags, after which the Go command is terminated.

After a breakpoint has been encountered, if you type the Go command again, then the program executes just as if you had typed the filename at the MS-DOS command level. The only difference is that program execution begins at the instruction after the breakpoint rather than at the usual start address.



## NAME

Hex

## PURPOSE

Performs hexadecimal arithmetic on the two parameters specified.

## SYNTAX

H&lt;value&gt; &lt;value&gt;

## COMMENTS

First, DEBUG adds the two parameters, then subtracts the second parameter from the first. The results of the arithmetic are displayed on one line; first the sum, then the difference.

## EXAMPLE

Assume that the following command is typed:

H19F 10A

DEBUG performs the calculations and then displays the result:

02A9 0095

## NAME

Input

## PURPOSE

Inputs and displays one byte from the port specified by <value>.  
(See note on page 2-37.)

## SYNTAX

I&lt;value&gt;

## COMMENTS

A 16-bit port address is allowed.

## EXAMPLE

Assume that you type the following command:

I2F8

Assume also that the byte at the port is 42H.  
DEBUG inputs the byte and displays the value:

42

## NAME

Load

## PURPOSE

Loads a file into memory.

## SYNTAX

L[&lt;address&gt; [&lt;drive&gt; &lt;record&gt; &lt;record&gt;]]

## COMMENTS

Set BX:CX to the number of bytes read. The file must have been named either when DEBUG was started or with the N command. Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the L command is typed without any parameters, DEBUG loads the file into memory beginning at address CS:100 and sets BX:CX to the number of bytes loaded. If the L command is typed with an address parameter, loading begins at the memory <address> specified. If L is typed with all parameters, absolute disk sectors are loaded, not a file. The <record>s are taken from the <drive> specified (the drive designation is numeric here--0=A:, 1=B:, 2=C:, etc.); DEBUG begins loading with the first <record> specified, and continues until the number of sectors specified in the second <record> have been loaded.

## EXAMPLE

Assume that the following commands are typed:

```
A>DEBUG
-NFILE.COM
```

Now, to load FILE.COM, type:

```
L
```

DEBUG loads the file and then displays the DEBUG prompt. Assume that you want to load only portions of a file or certain records from a disk. To do this, type:

```
L04BA:100 2 0F 6D
```

DEBUG then loads 109 (6D hex) records beginning with logical record number 15 into memory

beginning at address 04BA:0100. When the records have been loaded, DEBUG simply returns the - prompt.

If the file has a .EXE extension, it is relocated to the load address specified in the header of the .EXE file: the <address> parameter is always ignored for .EXE files. The header itself is stripped off the .EXE file before it is loaded into memory. Thus the size of an .EXE file on disk will differ from its size in memory.

If the file named by the Name command or specified when DEBUG is started is a .HEX file, then typing the L command with no parameters causes DEBUG to load the file beginning at the address specified in the .HEX file. If the L command includes the option <address>, DEBUG adds the <address> specified in the L command to the address found in the .HEX file to determine the start address for loading the file.

## NAME

Move

## PURPOSE

Moves the block of memory specified by <range> to the location beginning at the <address> specified.

## SYNTAX

M&lt;range&gt; &lt;address&gt;

## COMMENTS

Overlapping moves (i.e., moves where part of the block overlaps some of the current addresses) are always performed without loss of data. Addresses that could be overwritten are moved first. The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block's lowest address and then to work towards the highest. The sequence for moves from lower addresses to higher addresses is to move the data beginning at the block's highest address and to work towards the lowest.

Note that if the addresses in the block being moved will not have new data written to them, the data there before the move will remain. The M command copies the data from one area into another, in the sequence described, and writes over the new addresses. This is why the sequence of the move is important.

## EXAMPLE

Assume that you type:

```
MCS:100 110 CS:500
```

DEBUG first moves address CS:110 to address CS:510, then CS:10F to CS:50F, and so on until CS:100 is moved to CS:500. You should type the D command, using the <address> typed for the M command, to review the results of the move.

## NAME

Name

## PURPOSE

Sets filenames.

## SYNTAX

N&lt;filename&gt;[&lt;filename&gt;...]

## COMMENTS

The Name command performs two functions. First, Name is used to assign a filename for a later Load or Write command. Thus, if you start DEBUG without naming any file to be debugged, then the N<filename> command must be typed before a file can be loaded. Second, Name is used to assign filename parameters to the file being debugged. In this case, Name accepts a list of parameters that are used by the file being debugged.

These two functions overlap. Consider the following set of DEBUG commands:

```
-NFILE1.EXE  
-L  
-G
```

Because of the effects of the Name command, Name will perform the following steps:

1. (N)ame assigns the filename FILE1.EXE to the filename to be used in any later Load or Write commands.
2. (N)ame also assigns the filename FILE1.EXE to the first filename parameter used by any program that is later debugged.
3. (L)oad loads FILE1.EXE into memory.
4. (G)o causes FILE1.EXE to be executed with FILE1.EXE as the single filename parameter (that is, FILE1.EXE is executed as if FILE1.EXE had been typed at the command level).

A more useful chain of commands might look like this:

```
-NFILE1.EXE
-L
-NFILE2.DAT FILE3.DAT
-G
```

Here, Name sets FILE1.EXE as the filename for the subsequent Load command. The Load command loads FILE1.EXE into memory, and then the Name command is used again, this time to specify the parameters to be used by FILE1.EXE. Finally, when the Go command is executed, FILE1.EXE is executed as if FILE1 FILE2.DAT FILE3.DAT had been typed at the MS-DOS command level. Note that if a Write command were executed at this point, then FILE1.EXE--the file being debugged--would be saved with the name FILE2.DAT! To avoid such undesired results, you should always execute a Name command before either a Load or a Write.

There are four regions of memory that can be affected by the Name command:

```
CS:5C FCB for file 1
CS:6C FCB for file 2
CS:80 Count of characters
CS:81 All characters typed
```

A File Control Block (FCB) for the first filename parameter given to the Name command is set up at CS:5C. If a second filename parameter is typed, then an FCB is set up for it beginning at CS:6C. The number of characters typed in the Name command (exclusive of the first character, "N") is given at location CS:80. The actual stream of characters given by the Name command (again, exclusive of the letter "N") begins at CS:81. Note that this stream of characters may contain switches and delimiters that would be legal in any command typed at the MS-DOS command level.

#### EXAMPLE

A typical use of the Name command is:

```
DEBUG PROG.COM
-NPARAM1 PARAM2/C
-G
-
```

In this case, the Go command executes the file in memory as if the following command line had been typed:

PROG PARAM1 PARAM2/C

Testing and debugging therefore reflect a normal runtime environment for PROG.COM.



## NAME

Output

## PURPOSE

Sends the <byte> specified to the output port specified by <value>.  
(See note on page 2-37.)

## SYNTAX

O&lt;value&gt; &lt;byte&gt;

## COMMENTS

A 16-bit port address is allowed.

## EXAMPLE

Type:

O2F8 4F

DEBUG outputs the byte value 4F to output port 2F8.

## NAME

PTrace

## PURPOSE

Executes the instruction of the given start-address then displays the current value of all registers and flags. (See note on page 2-37.)

## SYNTAX

P[=start-address][count]

## COMMENTS

The display has the same format as the Register command. If the optional start-address is given, the command starts execution at the given address. Otherwise, it starts execution at the instruction pointed to by the current CS and IP registers. The equal sign (=) may be used only if a start-address is given.

If the optional count is given, the command continues to execute count instructions before stopping. The command displays the current value of the registers and flags for each instruction before executing the next.

The PTrace command is identical to the Trace command, except that it automatically executes and returns from any calls or software interrupts it encounters. The Trace command always stops after executing the call or interrupt, leaving execution control inside the called routine.

## EXAMPLE

Type:

P

DEBUG executes the instruction pointed to by the current CS and IP register values.

If you type

P=\_main

DEBUG executes the instruction at "\_main" then displays the current values of the registers and flags. It also displays the next instruction to be executed.

## NAME

Quit

## PURPOSE

Terminates the DEBUG utility.

## SYNTAX

Q

## COMMENTS

The Q command takes no parameters and exits DEBUG without saving the file currently being operated on. You are returned to the MS-DOS command level.

## EXAMPLE

To end the debugging session, type:

Q<RETURN>

DEBUG has been terminated, and control returns to the MS-DOS command level.

## NAME

Register

## PURPOSE

Displays the contents of one or more CPU registers.

## SYNTAX

R[&lt;register-name&gt;]

## COMMENTS

If no <register-name> is typed, the R command dumps the register save area and displays the contents of all registers and flags.

If a register name is typed, the 16-bit value of that register is displayed in hexadecimal, and then a colon appears as a prompt. You then either type a <value> to change the register, or simply press the <RETURN> key if no change is wanted.

The only valid <register-name>s are:

AX	BP	SS	
BX	SI	CS	
CX	DI	IP	(IP and PC both refer
DX	DS	PC	to the Instruction
SP	ES	F	Pointer.)

Any other entry for <register-name> results in a BR Error message.

If F is entered as the <register-name>, DEBUG displays each flag with a two-character alphabetic code. To alter any flag, type the opposite two-letter code. The flags are either set or cleared.

The flags are listed below with their codes for SET and CLEAR:

FLAG NAME	SET	CLEAR
Overflow	OV	NV
Direction	DN Decrement	UP Increment
Interrupt	EI Enabled	DI Disabled
Sign	NG Negative	PL Plus
Zero	ZR	NZ
Auxiliary Carry	AC	NA
Parity	PE Even	PO Odd
Carry	CY	NC

Whenever you type the command RF, the flags are displayed in the order shown above in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a hyphen (-). You may enter new flag values as alphabetic pairs. The new flag values can be entered in any order. You do not have to leave spaces between the flag entries. To exit the R command, press the <RETURN> key. Flags for which new values were not entered remain unchanged.

If more than one value is entered for a flag, DEBUG returns a DF Error message. If you enter a flag code other than those shown above, DEBUG returns a BF Error message. In both cases, the flags up to the error in the list are changed; flags at and after the error are not.

At startup, the segment registers are set to the bottom of free memory, the Instruction Pointer is set to 0100H, all flags are cleared, and the remaining registers are set to zero.

## EXAMPLE

Type:

R

DEBUG displays all registers, flags, and the decoded instruction for the current location. If the location is CS:11A, then the display will look similar to this:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000
SI=005C DI=0000 DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A NV UP DI NG NZ AC PE NC
04BA:011A CD21 INT 21
```

If you type:

RF

DEBUG will display the flags:

```
NV UP DI NG NZ AC PE NC - _
```

Now, type any valid flag designation, in any order, with or without spaces.

For example:

```
NV UP DI NG NZ AC PE NC - PLEICY<RETURN>
```

DEBUG responds only with the DEBUG prompt. To see the changes, type either the R or RF command:

RF

```
NV UP EI PL NZ AC PE CY - _
```

Press <RETURN> to leave the flags this way, or to specify different flag values.

## NAME

Search

## PURPOSE

Searches the <range> specified for the <list> of bytes specified.

## SYNTAX

S&lt;range&gt; &lt;list&gt;

## COMMENTS

The <list> may contain one or more bytes, each separated by a space or comma. If the <list> contains more than one byte, only the first address of the byte string is returned. If the <list> contains only one byte, all addresses of the byte in the <range> are displayed.

## EXAMPLE

If you type:

SCS:100 110 41

DEBUG will display a response similar to this:

04BA:0104  
04BA:010D  
-type:

## NAME

Trace

## PURPOSE

Executes one instruction and displays the contents of all registers and flags, and the decoded instruction.

## SYNTAX

T[=&lt;address&gt;][ &lt;value&gt;]

## COMMENTS

If the optional =<address> is typed, tracing occurs at the =<address> specified. The optional <value> causes DEBUG to execute and trace the number of steps specified by <value>.

The T command uses the hardware trace mode of the 8086 or 8088 microprocessor. Consequently, you may also trace instructions stored in ROM (Read Only Memory).

## EXAMPLE

Type:

T

DEBUG returns a display of the registers, flags, and decoded instruction for that one instruction. Assume that the current position is 04BA:011A; DEBUG might return the display:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000
SI=005C DI=0000 DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A NV UP DI NG NZ AC PE NC
04BA:011A CD21 INT 21
```

If you type

T=011A 10



DEBUG executes sixteen (10 hex) instructions beginning at 011A in the current segment, and then displays all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed. Then the display stops, and you can see the register and flag values for the last few instructions performed. Remember that <CONTROL-S> suspends the display at any point, so that you can study the registers and flags for any instruction.

## NAME

Unassemble

## PURPOSE

Disassembles bytes and displays the source statements that correspond to them, with addresses and byte values.

## SYNTAX

U[&lt;range&gt;]

## COMMENTS

The display of disassembled code looks like a listing for an assembled file. If you type the U command without parameters, 20 hexadecimal bytes are disassembled at the first address after that displayed by the previous Unassemble command. If you type the U command with the <range> parameter, then DEBUG disassembles all bytes in the range. If the <range> is given as an <address> only, then 20H bytes are disassembled instead of 80H.

## EXAMPLE

Type:

U04BA:100 L10

DEBUG disassembles 16 bytes beginning at address 04BA:0100:

04BA:0100	206472	AND	[SI+72],AH
04BA:0103	69	DB	69
04BA:0104	7665	JBE	016B
04BA:0106	207370	AND	[BP+DI+70],DH
04BA:0109	65	DB	65
04BA:010A	63	DB	63
04BA:010B	69	DB	69
04BA:010C	66	DB	66
04BA:010D	69	DB	69
04BA:010E	63	DB	63
04BA:010F	61	DB	61

If you type

U04ba:0100 0108

The display will show:

04BA:0100	206472	AND	[SI+72],AH
04BA:0103	69	DB	69
04BA:0104	7665	JBE	016B
04BA:0106	207370	AND	[BP+DI+70],DH

If the bytes in some addresses are altered, the disassembler alters the instruction statements. The U command can be typed for the changed locations, the new instructions viewed, and the disassembled code used to edit the source file.

## NAME

Write

## PURPOSE

Writes the file being debugged to a disk file.

## SYNTAX

W[&lt;address&gt;[ &lt;drive&gt; &lt;record&gt; &lt;record&gt;]]

## COMMENTS

If you type W with no parameters, BX:CX must already be set to the number of bytes to be written; the file is written beginning from CS:100. If the W command is typed with just an address, then the file is written beginning at that address. If a G or T command has been used, BX:CX must be reset before using the Write command without parameters. Note that if a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file (as long as the length has not changed).

The file must have been named either with the DEBUG invocation command or with the N command (refer to the Name command earlier in this manual). Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the W command is typed with parameters, the write begins from the memory address specified; the file is written to the <drive> specified (the drive designation is numeric here--0=A:, 1=B:, 2=C:, etc.); DEBUG writes the file beginning at the logical record number specified by the first <record>; DEBUG continues to write the file until the number of sectors specified in the second <record> have been written.

## WARNING

Writing to absolute sectors is EXTREMELY dangerous because the process bypasses the file handler.

EXAMPLE

Type:

W

DEBUG will write the file to disk and then display the DEBUG prompt. Two examples are shown below.

W

-

WCS:100 1 37 2B

DEBUG writes out the contents of memory, beginning with the address CS:100 to the disk in drive B:. The data written out starts in disk logical record number 37H and consists of 2BH records. When the write is complete, DEBUG displays the prompt:

WCS:100 1 37 2B

-

### 2.3 NOTES FOR PCW-1 MS-DOS USER

1. In addition to 8086/8087/8088 mnemonics, those for 80186 can be assembled with the A (Assemble) command.
2. Adding "W" to the I (Input) or O (Output) commands allows those commands to handle a word unit (2 bytes). For example:  
  
    IW2F8,     OW2F8 4263
3. When the "P"Trace command is used with the interrupt or call subcommands, then all subroutines are skipped.

## 2.4 ERROR MESSAGES

During the DEBUG session, you may receive any of the following error messages. Each error terminates the DEBUG command under which it occurred, but does not terminate DEBUG itself.

ERROR CODE	DEFINITION
BF	<p>Bad flag</p> <p>You attempted to alter a flag, but the characters typed were not one of the acceptable pairs of flag values. See the Register command for the list of acceptable flag entries.</p>
BP	<p>Too many breakpoints</p> <p>You specified more than ten breakpoints as parameters to the G command. Retype the Go command with ten or fewer breakpoints.</p>
BR	<p>Bad register</p> <p>You typed the R command with an invalid register name. See the Register command for the list of valid register names.</p>
DF	<p>Double flag</p> <p>You typed two values for one flag. You may specify a flag value only once per RF command.</p>

## INDEX

## DEBUG Commands

(A)ssemble . . . . .	2-6
(C)ompare . . . . .	2-8
(D)ump . . . . .	2-9
(E)nter . . . . .	2-11
(Fill). . . . .	2-13
(G)o . . . . .	2-14
(H)ex . . . . .	2-16
(I)nput . . . . .	2-17
(L)oad . . . . .	2-18
(M)ove . . . . .	2-19
(N)ame . . . . .	2-21
(O)utput . . . . .	2-24
(P)Trace. . . . .	2-25
(Q)uit . . . . .	2-26
(R)egister. . . . .	2-27
(S)earch . . . . .	2-30
(T)race . . . . .	2-31
(U)nassemble . . . . .	2-33
(W)rite . . . . .	2-35
DEBUG Errors	
BF - Bad flag . . . . .	2-38
BP - Too many breakpoints . . . . .	2-38
BR - Bad register . . . . .	2-38
DF - Double flag . . . . .	2-38
EXE files . . . . .	2-19
Flags . . . . .	2-28



© Copyright MINOLTA CAMERA CO., LTD. 1985. All rights reserved.

© Copyright Microsoft Corporation, 1982, 1983.

IBM is a registered trademark of International Business  
Machine Corporation.

Microsoft is a registered trademark of Microsoft Corporation.

MS is a trademark of Microsoft Corporation.

1-2-3 and Lotus are trademarks of Lotus Development Corporation.