

MSTM-DOS Operating System

Software Library Manual



Texas Instruments Professional Computer

Copyright © 1982 by Texas Instruments Incorporated
Portions of this manual are reproduced by permission of the
copyright owner, Microsoft Corporation

MS™-DOS Operating System
TI Part No. 2223133-0001
Original Issue: 10 December 1982

THIS PACKAGE CONTAINS SOFTWARE COPYRIGHTED BY MICROSOFT CORPORATION AND TEXAS INSTRUMENTS INCORPORATED. THIS PACKAGE IS DISTRIBUTED BY TEXAS INSTRUMENTS. YOU SHOULD CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE USING THE SOFTWARE. USE OF THE SOFTWARE INDICATES YOUR ACCEPTANCE OF THE TERMS AND CONDITIONS.

UNDER THE COPYRIGHT LAWS THERE ARE PENALTIES FOR MAKING UNAUTHORIZED COPIES. YOU MAY MAKE TWO COPIES IN MACHINE-READABLE FORM FOR YOUR OWN USE FOR BACKUP AND ARCHIVAL PURPOSES. ANY COPY YOU MAKE MUST INCLUDE REPRODUCTION OF THE COPYRIGHT NOTICE. YOU MAY NOT SELL OR OTHERWISE TRANSFER ANY COPIES THAT YOU MAKE.

MICROSOFT CORPORATION AND TEXAS INSTRUMENTS MAKE NO EXPRESS OR IMPLIED WARRANTY FOR THE SOFTWARE WHICH IS PROVIDED SOLELY ON AN "AS IS" BASIS. MICROSOFT AND TEXAS INSTRUMENTS DISCLAIM ALL CONDITIONS AND WARRANTIES, WHETHER EXPRESS OR IMPLIED WITH REGARD TO THE SOFTWARE, INCLUDING ALL IMPLIED CONDITIONS OR WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER OF THEM SHALL HAVE ANY LIABILITY OR RESPONSIBILITY OF ANY KIND, INCLUDING SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, ARISING OUT OF OR RESULTING FROM SUCH SOFTWARE OR THE USE OR MODIFICATION THEREOF.

Preface

MSTMDOS stands for Microsoft Disk Operating System. Just as businesses need managers to organize and allocate the company's resources, your Texas Instruments Professional Computer needs a manager to oversee and direct its various components. MS-DOS serves as that manager.

MS-DOS manages the information stored on your computer and directs the activities of its components. Using a few simple MS-DOS commands you can:

- Manage information
- Modify text
- Run programs
- And in general, simplify the use of your computer

You don't have to have experience using computers to be able to use MS-DOS. It is important, however, that you are familiar with the material presented in the *Texas Instruments Professional Computer Operating Instructions* (TI Part No. 2223116-0001), especially Chapter 3, "Using MS-DOS."

USING THIS MANUAL

The best way to learn about MS-DOS is to use it. Try things with your Texas Instruments Professional Computer as you read through this manual. MS-DOS is not at all difficult to use—it simply takes practice.

This section gives you a brief overview of the manual. We suggest that you read the first four chapters no matter what your level of computer experience, just to acquaint yourself with using MS-DOS on a Texas Instruments Professional Computer.

Chapter 1, “System Start-up,” gets you started using MS-DOS. Remember, we are assuming you have already worked through the “Using MS-DOS” section of the *Operating Instructions*. This chapter is just a brief review.

Chapter 2, “Working With MS-DOS,” discusses the use of files and commands. In Chapter 2 you learn to name files and to use and interpret commands. This chapter not only tells you what happens, but also tells you a little about how it happens. Reading through this chapter will give you a better idea of how MS-DOS works.

Chapter 3, “Command Directory,” is an alphabetical listing of the MS-DOS commands with complete explanations and helpful examples.

Chapter 4, “Line Editor,” describes the MS-DOS line editor, EDLIN. EDLIN is an MS-DOS program which makes it easy for you to edit your files.

Chapter 5 and 6, “Debugging (DEBUG)” and “File Comparison (FILCOM),” explain the MS-DOS debugger and file comparison utilities. While Chapter 6 is useful to both experienced and inexperienced computer users, Chapter 5 will be useful to you only if you intend to create computer programs.

In the back of this manual are several appendixes. If you have a single-diskette-drive system, you will want to refer to Appendix A, “Single-Diskette Drive Users.” Appendix D, “MS-DOS Messages,” is helpful in explaining any error messages you may receive. Experienced programmers will find Appendix B, “MS-DOS File-Control-Block Definition” and Appendix C, “Interrupts and Function Calls” useful.

STORING INFORMATION

Like any manager, MS-DOS needs to be able to organize and store information on files. The files MS-DOS uses have an advantage over traditional paper files in that they take up less room, are convenient to access, and are easy to modify. All the information on the MS-DOS diskette is stored in files.

In working with MS-DOS, you will use your own files frequently to store and recall the information you keep on your computer. Each file has a unique name and location on your diskette.

These files are organized collectively into a “directory.” The directory contains information about each file. This information includes the complete filename, the size, and the time and date of the last modification.

Both files and directories take up space on the diskette. As information is placed on the diskette, it is given a specific location so the computer can find it the next time it is needed. Diskette space is divided into a number of concentric tracks much like a phonograph record. Each track is then divided into a number of sectors. The computer assigns a specific location on the diskette with a track number and a sector number.

COMMANDS

Now that you have some idea of how MS-DOS stores and accesses information, you may wonder what part you play. While MS-DOS may be an efficient operating system, you are still the operator. MS-DOS can’t do anything without your instructions. You have to tell the operating system what to do.

You instruct MS-DOS through commands. When you enter a command it appears on the display unit. The line on which it appears is referred to as the command line. MS-DOS provides special functions that allow you to edit the command line easily. These special functions and more details about using commands are discussed in Chapter 2.

Chapter 3 contains a directory of the MS-DOS commands arranged in alphabetical order. Here you will find a complete explanation of each command along with many practical examples.

You are now ready to start up the system and issue your first command to MS-DOS. Chapter 1 tells you how.

Contents

Preface	iii
1 System Start-up	1-1
Loading MS-DOS	1-3
Making Backups	1-4
Personalized Start-up	1-5
MS-DOS System Files	1-5
Syntax Notation	1-7
2 Working With MS-DOS	2-1
Introduction	2-3
Communication of Instructions	2-3
Naming Files	2-13
Wild-Card Characters	2-14
Command Interpretation	2-17
3 Command Directory	3-1
Check Disk (CHKDSK)	3-5
Configure (CONFIG)	3-8
COPY	3-12
COPY with Concatenation	3-16
DATE	3-20
Delete (DEL)	3-21
Directory (DIR)	3-22
Diskette Compare (DISKCOMP)	3-23
Diskette Copy (DISKCOPY)	3-26
ERASE	3-28
EXE to Binary (EXE2BIN)	3-29
FORMAT	3-31
PAUSE	3-33
Remark (REM)	3-34
Rename (REN)	3-34
System (SYS)	3-35

TIME	3-36
TYPE	3-38
MS-DOS Commands	3-38

4 Line Editor 4-1

Terms	4-3
Starting EDLIN	4-4
Intraline Commands	4-9
Interline Commands	4-19
Errors When Invoking EDLIN	4-47
Errors While Editing	4-48

5 Debugging (DEBUG) 5-1

Invocation	5-3
Commands	5-5
Parameters	5-5
COMPARE	5-10
DUMP	5-10
ENTER	5-12
FILL	5-14
GO	5-15
HEX	5-16
INPUT	5-17
LOAD	5-17
MOVE	5-19
NAME	5-20
OUTPUT	5-23
QUIT	5-24
REGISTER	5-24
SEARCH	5-27
TRACE	5-28
UNASSEMBLE	5-29
WRITE	5-31
Command Summary	5-33
Error Messages	5-34

6 File Comparison (FILCOM)	6-1
Limitations on Source Comparisons	6-3
Invocation	6-3
Commands	6-4
Examples	6-12

Appendixes

A Single Diskette-Drive Users

B File-Control-Block Definition

C Interrupts and Function Calls

D MS-DOS Messages

Index

1

System Start-up

Loading MS-DOS	1-3
Making Backups	1-4
Personalized Start-up	1-5
MS-DOS System Files	1-5
Syntax Notation	1-7

LOADING MS-DOS

This chapter gives a very brief overview of how to start up MS-DOS. At this point if you are not familiar with how to install the MS-DOS diskette and make a backup diskette for your own use, you should refer to Chapter 3 of the *Operating Instructions*, “Using MS-DOS,” for instructions.

Start or “boot” MS-DOS by inserting the MS-DOS diskette. Then place the system unit ON/OFF switch in the ON position. If your system is already on, you can warm-start MS-DOS by pressing and holding the **CTRL** and **ALT** keys then pressing the **DEL** key. This assumes your MS-DOS diskette is already inserted in one of the diskette drives.

When your system finishes loading MS-DOS, it displays a sign-on message containing the version numbers. It then gives you the current date followed by the prompt:

Enter new date : []

If you don’t want to change the date, simply press the **RETURN** key. To enter a new date, use the format month-day-year (mm-dd-yy) where

mm = 1–12

dd = 1–31

yy = 80–99 or 1980–1999

For example, entering:

8-3-82

tells the computer that today is August 3, 1982.

You must separate the parts of the date with either a hyphen (-) or a slash (/).

The next prompt displays the current time and asks you to enter a new time. As in the case of the current date, if you don’t want to enter a new time, press the **RETURN** key.

If you wish to enter a new time, use the format hour:minute:seconds.hundredths, where:

(hh : mm : ss. hh)

hh = 00–24

mm = 00–59

ss = 00–59

hh = 00–99

For example, entering:

8:20:00

tells the computer that the current time is 8:20:00 A.M. You specify 2:30 P.M. by entering:

14:30

Once you have set the date and time or pressed the **RETURN** key, the system prompt appears. This prompt indicates that your computer is ready to accept your commands.

MAKING BACKUPS

If you have not formatted a new diskette and copied the original MS-DOS diskette onto it, do so now. You can refer to Chapter 3 of the *Operating Instructions* for additional information on these procedures.

You should always work from backup diskettes. If your original diskette gets damaged or if it becomes worn from heavy use, the only way to replace it is to buy a new one.

In addition to backing up your MS-DOS diskette, you should make frequent backups of your own work. This allows you to keep earlier versions of your files. More importantly, a recent backup diskette can prove invaluable if your current diskette is lost or damaged.

PERSONALIZED START-UP

Whenever you start up, or boot, MS-DOS, the operating system is automatically loaded from the diskette into memory. MS-DOS is loaded into an area referred to as the system memory.

The first thing MS-DOS does after it has been loaded from the diskette is look for a file named AUTOEXEC.BAT. This filename refers to a batch file which (if it has been created) is automatically executed every time the system is started up.

A batch file is simply a file that contains a set of commands that are automatically executed in order, one at a time, just as if you were typing them at your computer.

AUTOEXEC.BAT does not exist unless you create it. It is an optional file. If you do decide to define an AUTOEXEC.BAT file, the date and time prompts will not be shown when the system is started up.

For more information about AUTOEXEC.BAT and batch files in general, see the “Batch Commands” section of Chapter 2 in this manual.

MS-DOS SYSTEM FILES

MS-DOS consists of three files:

- COMMAND.COM
- MSDOS.SYS
- IO.SYS

These three files combine to form an operating system that controls all system resources.

Type the command

DIR

and press the **RETURN** key. Notice that MSDOS.SYS and IO.SYS are not displayed in the directory listing. These are “hidden” files that are put in a specific location on your diskette by the FORMAT command. You don’t have to be concerned with the function of these files.

Unlike the other two MS-DOS system files, COMMAND.COM appears in the directory listing. COMMAND.COM is the program responsible for interpreting and executing your instructions to MS-DOS.

COMMAND.COM provides the colon prompt (:) for MS-DOS. The prompt appears in the form of a drive designation letter and a colon (:).

For example:

A:[]

Assuming you boot from diskette drive A, at system start-up the default prompt is always A: . After start-up, you may specify the default (that is, the currently selected) drive. To select a new drive, simply enter the drive designation letter followed by a colon:

A:[]

you enter

B:

and press the **RETURN** key. The new prompt appears as:

B:[]

This program and its relationship to your instructions are discussed in more detail in Chapter 2, “Working With MS-DOS.”

SYNTAX NOTATION

The following 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 information that you enter. When the angle brackets enclose lowercase text, you must type in an entry defined by the text; for example, <filename>. When the angle brackets enclose uppercase 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.
- ... Ellipses points indicate that an entry may be repeated as many times as needed.

CAPS Capital letters indicate portions of statements or commands that must be entered exactly as shown. When capital letters appear within angle brackets, they indicate pressing of keys or key combinations such as <CTRL-C> or <RETURN>. <CTRL-C> indicates that you press and hold the **CTRL** key and then type a **C**.

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

The cursor is the focus of any editing you perform. In this manual the cursor is indicated as shown below:

[]

Working With MS-DOS

Introduction	2-3
Communication of Instructions	2-3
Command-Line Editing	2-3
Special Editing Keys	2-4
Naming Files	2-13
Wild-Card Characters	2-14
The ? Character	2-14
The * Character	2-15
Device Names	2-16
Command Interpretation	2-17
Command Types	2-19
Internal Commands	2-20
External Commands	2-20
Batch Commands	2-21
Dummy Parameter	2-23
AUTOEXEC. BAT File	2-24

INTRODUCTION

This chapter explains MS-DOS files and commands and discusses:

- Using and editing the command line
- Naming and using files
- Different types of commands
- When and where to use commands
- How to combine commands

COMMUNICATION OF INSTRUCTIONS

MS-DOS acts as a middleman between you and the resources of your system. You communicate with MS-DOS through instructions or commands typed on the keyboard. The line on which you type commands is referred to as the command line. MS-DOS has special functions you can use to edit this command line.

When you finish editing the command line, it is sent to the COMMAND.COM program to be interpreted and processed by MS-DOS. Thus, your communication with your computer has two levels—command-line editing, which you do, and command interpretation, which is done by MS-DOS. The first level of processing is discussed in the following section.

Command-Line Editing

The computer stores anything you type on a command line in a temporary storage area. This area is called the command-line buffer. Until you enter your command by pressing the **RETURN** key you can change the command line. All your modifications are stored in the command-line buffer.

When you are satisfied with your instructions and want the computer to process them, simply press the **RETURN** key. When you press the **RETURN** key two things happen:

1. The contents of the command line are sent to **COMMAND.COM** for processing.
2. The command-line buffer is copied into another temporary storage area called the “template.”

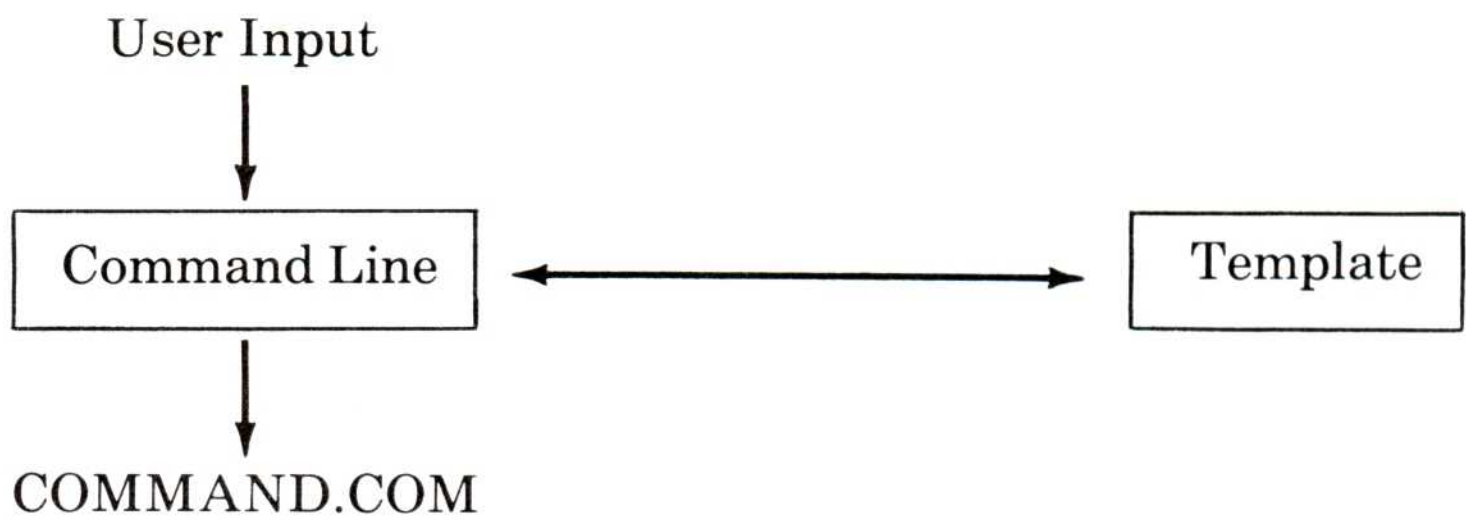
This template always contains the last command you entered. MS-DOS provides functions that can copy all or part of your last command into your new command line. Once you are familiar with the command-line editing keys, you’ll find they save you a great deal of work.

Special Editing Keys

The special editing keys allow you to avoid retyping a sequence of keys because MS-DOS automatically remembers the last command line entered and places it in the template. The advantages of using the special editing keys include:

- A command line can be repeated in two keystrokes.
- A command line which contains an error can be edited and retried, without retyping the entire command line.
- A command line similar to the preceding command line can be edited and executed with a minimum of typing.

The following illustration shows the relationship between the command line and the template.



The table below contains a complete list of the special editing keys. These special editing keys are a subset of the functions available within the MS-DOS text editor and are described more fully in Chapter 4, “Line Editor.” When editing text, the special editing keys are referred to as “intraline editing commands.”

Key	Function
F2	Copy all characters up to the character specified from the template to the command line.
F3	Copy all remaining characters in the template to the command line.
F4	Skip over (do not copy) the characters in the template up to the character specified.
F5	Make the new line the new template. An @ sign appears to show that this has occurred.

Key	Function
RIGHT ARROW	Copy one character from the template to the command line.
LEFT ARROW	Backspace. Erases the previous character from the command line and the display.
DOWN ARROW	Stop current command line without changing the template.
INS (Insert)	Enter insert mode.
DEL (Delete)	Skip over (do not copy) a character in the template.

Once you become accustomed to using them, the MS-DOS special editing keys are great timesavers. They are used often in editing both the command line and lines of text. The following is a hands-on practice session demonstrating the use of these keys in editing the command line. You should follow this section using your Texas Instruments Professional Computer.

Begin by typing the command:

This is a sample command.

Now press the **F5** key. Your display should look like this:

```
< F5>  A: This is a sample command.@
        [ ]
```

In this manual the keys you press are indicated to the left of the sample command line. This information is there to illustrate our examples. Nothing actually appears to the left of the sample command line on your display. Pressing the **F5** key copied the command from the command-line buffer into the template. An @ sign at the end of the command indicates that this has happened.

Now try pressing the **RIGHT ARROW** key. This copies one character from the first command to the new command line.

```
A:This is a sample command.@  
< RIGHT ARROW> T[]
```

Try pressing the **RIGHT ARROW** key several times. Each time the **RIGHT ARROW** key is pressed, one more character appears.

```
< RIGHT ARROW> Th[]  
< RIGHT ARROW> Thi[]  
< RIGHT ARROW> This[]
```

Continue pressing the **RIGHT ARROW** key until the command you typed initially appears in its entirety. Press the **F5** key. Your display should look like this:

```
      This is a sample command.@  
< F5> []
```

You can copy all the characters up to a specified character from the first command by pressing the **F2** key. Try pressing the **F2** key and then typing the letter **p**. Your display should look like this:

```
      This is a sample command.@  
< F2> p This is a sam[]
```

To copy all the remaining characters from the template press the **F3** key. Then press the **F5** key to make the new line the new template.

```
< F3>      This is a sample command.@  
< F5>      []
```

To skip over a character in the template, press the **DEL** key. Each time you press the **DEL** key, one character is deleted (not copied from the template onto the input buffer). Try pressing the **DEL** key. Your display should look like this:

```
      This is a sample command.  
< DEL> []
```

Note that the cursor does not move. Only the template has been affected. To see how much of the command has been skipped, press the **F3** key to copy the remaining characters from the template to the input buffer.

```
      This is a sample command.  
< DEL> []  
< F3>  his is a sample command.[]
```

Suppose now you wanted to return the command to its original condition. You need to insert the letter T at the beginning of the command. First press the **F5** key to make your display look like this:

```
      his is a sample command.@  
< F5> []
```

Press the **INS** key. You are now in the insert mode. Anything you type will be inserted at the front of the command currently in the template. Type the letter **T**. Your display should look like this:

```
      his is a sample command.@  
< INS> T T[]
```

To copy the remaining characters from the template and leave the insert mode, press the **F3** key.

```
      This is a sample command.@  
< INS> T[]  
< F3> This is a sample command.[]
```

Suppose you wanted to skip over several characters in the template. You can skip over all the characters in the template up to a specified character by pressing the **F4** key and typing a character. For example, assume you have pressed the **F5** key, and your display looks like this:

```
This is a sample command.@  
[ ]
```

Press the **F4** key and then type the letter **p**. Your display should now look like this:

```
                This is a sample command.@  
< F4> p [ ]
```

Note that the cursor does not move. To find out how many characters have been skipped, press the **F3** key to copy all the remaining characters in the template.

```
                This is a sample command.@  
< F4> p [ ]  
< F3>  ple command.[ ]
```

Restore your original command by entering the insert mode and inserting the necessary characters as demonstrated previously, pressing the **F5** key to restore the template. Your display should now look like this:

```
                This is a sample command.@  
< F5> [ ]
```

Use the **DEL** key followed by the **F3** key to skip over a character in the template then display the current command line.

```
                This is a sample command.@  
< DEL> < F3> his is a sample command.[ ]
```

Instead of pressing the **RETURN** key which would execute the command or pressing the **F5** key which would make the new line the new template, try pressing the **DOWN ARROW** key instead. Your display should look like this:

```
< DEL> < F3>      This is a sample command.@
< DOWN ARROW>    his is a sample command.\
                  [ ]
```

The **DOWN ARROW** key outputs a backslash (\) and voids the current input. The template has not changed. To restore the line simply press the **F3** key.

Practice using these keys in different combinations on your own. Remember the best way to learn to apply MS-DOS and its features to your own work is to use it. Don't be afraid to experiment.

You may also want to use control keys in your work. The following table lists these keys and their equivalent control-characters with a brief description of their function. To type a control character simply press and hold the **CTRL** key while you type the character. Since it is much faster and easier to use the control keys, control characters are used less often.

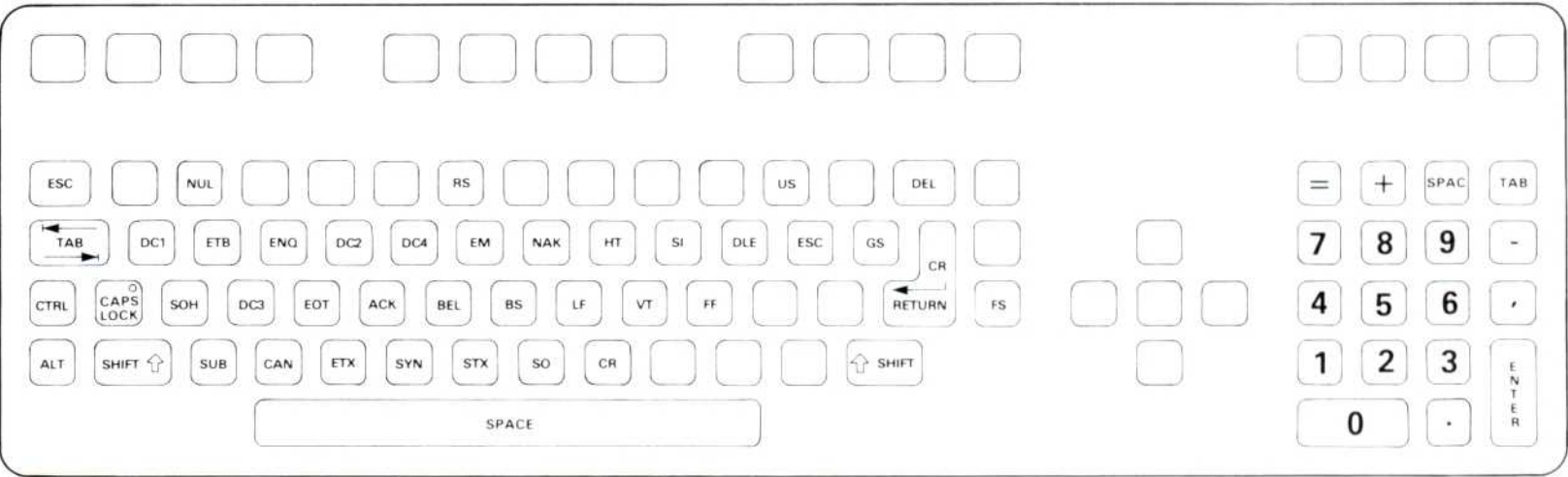
Control Key	Alternate Control Character	Function
PRNT	< CTRL-P> * or < CTRL-N>	Begin echoing of output to the printer. (Pressing the PRNT key or the < CTRL-N> again cancels the echoing.)
SHIFT/BRK**	< CTRL-C>	Abort the current operation.
LEFT ARROW or BACKSPACE	< CTRL-H>	Remove the last character from the command line, and erase the character from the display.
Line Feed	< CTRL-J>	Insert a physical end-of-line, but do not empty the command line. Use a linefeed to extend the current line beyond the physical limits of one display line.
(no key)	< CTRL-Z>	End-of-String, end of insert in EDLIN.
BRK/PAUS	< CTRL-S>	Suspend display of output to the display unit. Press any key (including BRK/PAUS again) to resume.
DOWN ARROW	< CTRL-X>	Cancel the current line, empty the command line, and then output a backslash, carriage return, and linefeed. The template used by the special editing commands is not affected.

* < CTRL-P> means press and hold the **CTRL** key and then press the **P** key.

** SHIFT/BRK means press and hold the **SHIFT** key and then press the **BRK/PAUS** key.

Control Key	Alternate Control Character	Function
TAB	< CTRL-I >	Tab.
SHIFT/PRNT		Copy information currently shown on the display to the printer.
RETURN	< CTRL-M >	Send displayed line to requesting program.

In many cases pressing and holding the **CTRL** key and then pressing another key on your keyboard causes the function of that key to be altered. In this way several characters that do not appear on your keyboard can be produced. The following diagram shows the function of your keyboard keys if you pressed them while you were pressing and holding the **CTRL** key.



2223133-1

NAMING FILES

To name a file, you use a three-part name known as a file specification. A complete file specification, or a filespec, contains a drive designation, a filename, and a filename extension. The format of a file specification is:

d: filename. ext

A file specification is made up of the following parts.

The **d:** is the drive designation. If you want to name a file in the default drive, it's not necessary to use a drive designation.

The **filename** is the name you want to give your file. Filenames can be from one to eight characters. The characters that make up a filename can be:

- Any letter of the alphabet
- Any number (0–9)
- Any one of the following symbols:

\$	&	#	@	!		
%	'	()	-	<	>
\	^	{	}	~	,	;

You can use any of these characters in any combination in a filename. Lowercase letters may be used in your file specification, but the computer will automatically convert them to upper case. This means that “FiLe.ExT” is converted to “FILE.EXT”.

The **.ext** is the filename extension. It should consist of at most three characters and should always be preceded by a period. Any of the characters you can use in a filename can be used in a filename extension. As in a filename, all lowercase characters are automatically converted to upper case.

Extensions are usually optional. You can use them to make your filename more descriptive. For example, the filename:

SALES.82

tells you more about the file than the name SALES would.

Some commands operate on files which are assumed to have a default extension. In these cases you can either specify the filename with the required extension or simply the filename with a period after it.

WILD-CARD CHARACTERS

Two special characters, the asterisk (*) and the question mark (?), can be used in file specifications. We call these “wild-card characters” because they can take the place of any character or group of characters in a filename or filename extension.

You can use these special characters to access several files with only one file specification. This is particularly useful when you are using commands such as directory, delete, and copy.

The ? Character

When MS-DOS encounters a question mark (?) in a file specification, it searches for all file specifications which differ only by the character in the question-mark position. For example, the command:

DIR AB?DE.EXT

lists all the directory entries on the default drive with five character filenames that begin with AB, end with DE, and have the extension. EXT. The files listed by MS-DOS might be:

**ABCDE.EXT
ABODE.EXT
ABIDE.EXT**

The * Character

The asterisk (*) is shorthand for a series of question marks (?). When MS-DOS encounters an asterisk in a file specification, it searches for all file specifications which differ only by the characters in the asterisk position. For example, the following expressions are equivalent:

```
DIR *.*  
DIR ????????.???
```

If you typed the command,

```
DIR AB*.EXT
```

the files listed by MS-DOS might be:

```
ABCDE.EXT  
ABC123.EXT  
ABIDE.EXT  
ABO$$$.EXT  
ABODE.EXT
```

Suppose you wanted to list all the directory entries on drive A with the filename FILE, regardless of their extension. You could use either

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

To list all the directory entries for files with the extension .EXT on the default drive, regardless of their filenames, you can use either:

```
DIR *.EXT  
or  
DIR ????????.EXT
```

To delete all the files on drive B whose filenames begin with AB and that have an extension that begins with .E you could use either:

DEL B:AB?????.E??

or

DEL B:AB*.E*

You should exercise great caution when using wild-card characters in a command to delete a group of files. It is safer to delete files individually, without using wild-card characters. When using wild-card characters to delete files first use the directory command with the wild-card characters to list the files you intend to delete.

Device Filenames

MS-DOS refers to the various devices in your system by filenames also. This makes it easy for you to talk to a different part of your system—the printer for example—and tell it what to do.

Certain three-letter filenames are reserved for the names of devices. These filenames are listed below.

AUX	Refers to information exchanged between an auxiliary device and your computer.
CON	Refers to information exchanged between the keyboard and the display unit (console).
LST or PRN	Refers to the printer or list device.
NUL	Used when the format of a command requires a filename, but you do not wish to use a particular file.

Even if these filenames are given drive designations or extensions, they remain associated with the devices listed above. The file specification:

A:CON.LIST

still refers to the display unit and cannot be used as the name of a diskette file.

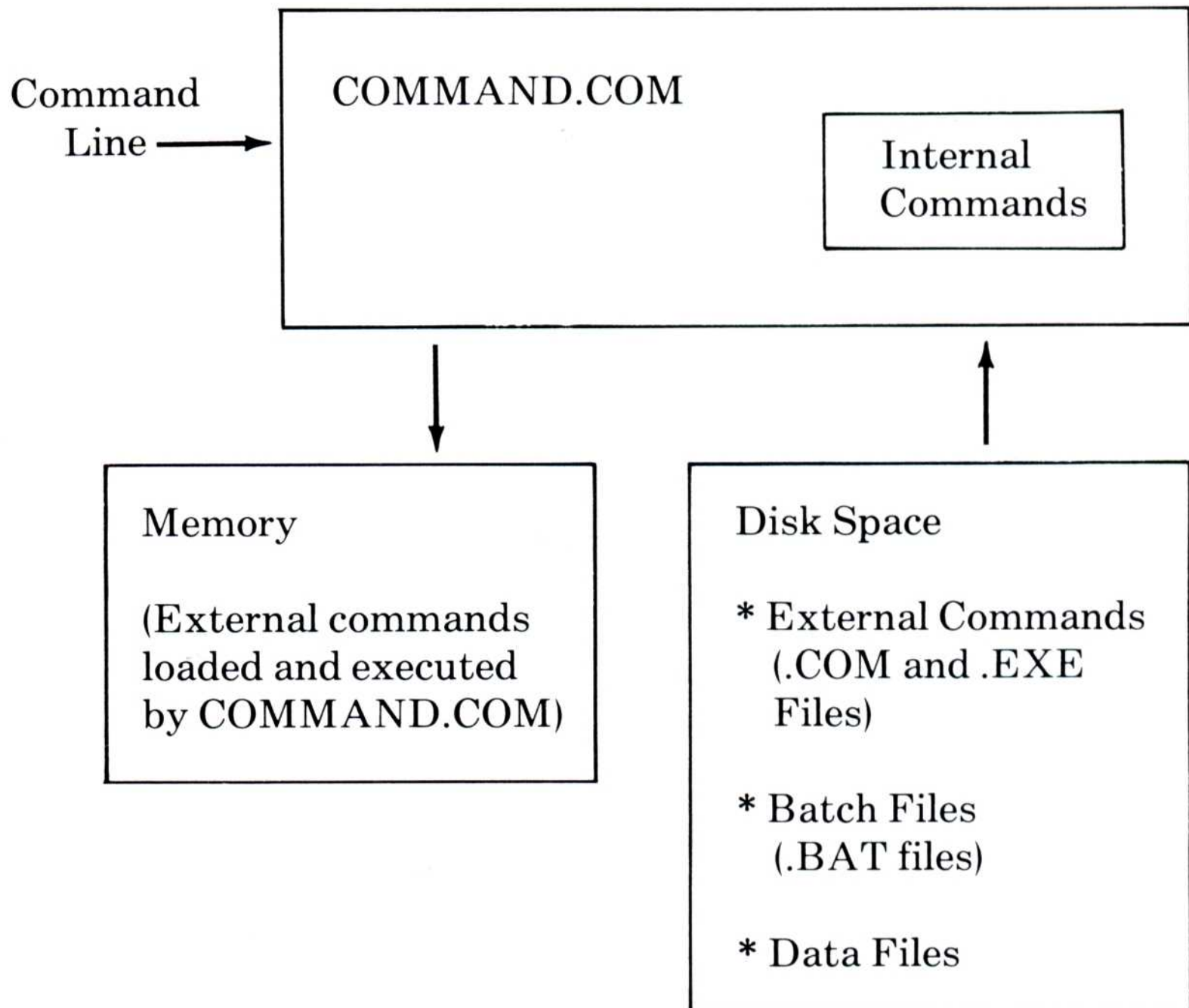
COMMAND INTERPRETATION

Earlier in this chapter we discussed how to edit the command line with special editing and control-character functions. This is the first level in the processing of your instructions. The next level, command interpretation, is handled by the COMMAND.COM program on your MS-DOS diskette.

Although COMMAND.COM takes care of the details, there are a few things about interpreting commands that you should know. This section briefly describes the function of the COMMAND.COM program and how it works with different types of commands.

Commands can be classified into two categories—those that are a permanent part of COMMAND.COM and those that COMMAND.COM loads from the diskette only when they are needed. Commands that are resident in memory as a part of the COMMAND.COM program are called internal commands. Commands that COMMAND.COM must load from the diskette into memory before processing are called external commands.

Internal commands are loaded into memory when the operating system is loaded. External commands are stored in diskette files which have either a .COM or an .EXE extension. Notice that COMMAND.COM is itself an external command. The following diagram shows the relationship among COMMAND.COM, the computer's memory, and diskette space.



Command interpretation begins when COMMAND.COM scans the command line for a legal command. If the command is internal, it is executed immediately. If the command is external, the appropriate file is loaded from diskette into memory where it is executed. The COMMAND.COM response to a batch file is discussed later in this chapter.

The primary function of COMMAND.COM is to identify commands and execute them. All commands consist of a command name followed by optional parameters.

Parameters give a command the conditions under which to operate. For example, you can use the command COPY to copy the file named OLDFILE.REL into another file named NEWFILE.REL.

COPY OLDFILE.REL, NEWFILE.REL

In the above example, the file specifications OLDFILE.REL and NEWFILE.REL are the parameters. They indicate which files the computer uses to perform the COPY command.

Parameters must be separated from the command name and from each other. Spaces, commas, semicolons, equal signs, and tabs are the only legal separators. We could have written our earlier example of the COPY command using a space instead of a comma between the parameters, in which case it would look like this:

COPY OLDFILE.REL NEWFILE.REL

Command Types

COMMAND.COM allows you to execute three different types of commands:

- Internal commands such as DIR, REN, TYPE, and DEL.
- External commands including .COM commands such as CHKDSK.COM, DEBUG.COM, and EDLIN.COM and .EXE commands such as LINK and EXE2BIN.
- Batch (.BAT) command files that contain multiple instances of the above commands.

The internal commands and most commonly used external commands are listed in alphabetical order with complete descriptions in Chapter 3, “Commands.” Study these commands carefully—you will use them frequently in your work.

Internal Commands

Internal commands are incorporated in COMMAND.COM and are always available when COMMAND.COM is resident in memory. Most internal commands are simple and easy to use.

The internal commands are listed below. Each is described in detail in the next chapter. They are:

COPY	REN
DATE	PAUSE
DEL	TIME
DIR	TYPE
REM	ERASE

External Commands

Any file with the extension .COM or .EXE is considered valid as an external command. Such commands are executed by entering the name of the file without the .COM or .EXE extension. In the event you create program files using computer languages, most of them will be .EXE files.

External files include:

CHKDSK	EDLIN
COMMAND	FORMAT
CONFIG	FILCOM
DEBUG	LINK
DISKCOMP	SYS
DISKCOPY	

NOTE FOR EXPERIENCED PROGRAMMERS:

An .EXE file created with the MACRO-86 assembler can be converted to a .COM file with the command EXE2BIN.EXE. The format of a .COM file is special, so a .EXE file cannot be arbitrarily converted. Note also that all .COM commands are executed in less than 64K bytes of memory. .EXE files, on the other hand, may require more than 64K bytes of memory for execution.

The next section of this chapter discusses the use of the MS-DOS batch facility. The batch facility allows you to combine a set of commands into a file and then execute the file using only one command.

Separate chapters are also provided on three utility programs. They are:

Chapter 4,	“Line Editor”
Chapter 5,	“Debugging”
Chapter 6,	“File Comparison”

Of these three, the chapters on line editing and file comparison are probably the ones you will probably find most useful if you do not intend to use your Texas Instruments Professional Computer to develop your own programs.

Batch Commands

The MS-DOS batch facility allows a set of commands contained in a file to be executed. A “batch” of commands in a file is processed as though each command were typed at the keyboard.

To use the batch facility, you must first create a file with a filename and .BAT extension. Creating a file is discussed in Chapter 4. Do not give a batch file the same name as a .EXE or .COM file. When you want to process a batch file, you refer to it by the filename only. Do not include the .BAT extension.

You may specify optional parameters. This way you can use your batch file with different sets of data on different occasions.

The format of a batch command is:

[d:] filename [parameters]

where the drive designation and parameter specifications are optional.

The following is a sample batch file as viewed from EDLIN, the MS-DOS line editor. Don't worry if the commands are unfamiliar to you. Chapter 3 discusses these commands in detail.

```
1:  REM This is file NEWDISK.BAT
2:  PAUSE Insert diskette in B:
3:  FORMAT B:/S
4:  DIR B:
5:  CHKDSK B:
```

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

NEWDISK

Press the **RETURN** key. The result is the same as if each of the commands in the .BAT file were entered at the keyboard as individual commands. Two MS-DOS commands are available expressly for their use in batch files—REM and PAUSE. REM permits you to include remarks and comments in batch files. PAUSE prompts you with a message (optional) and permits you either to continue or to abort execution of a batch file at a given point. These commands and the ones used in the examples above are discussed further in the next chapter.

Dummy Parameters

When you create a batch file you may want to make it general enough to be used with different sets of parameters. You can do this by putting in replaceable or “dummy” parameters. These dummy parameters are replaced by values you specify when the batch file is executed.

You may use up to ten dummy parameters, named %0 through %9. The %0 parameter is always replaced by the drive designator (if specified) and the filename of the batch file.

For example, the following batch file contains dummy parameters:

```
1:  REM This is ASMFIL.BAT
2:  COPY %1.ASM %2.ASM
3:  DEL %1.ASM
4:  TYPE %2.ASM
5:  TYPE %0.BAT
```

This set of commands copies one file into another file and then deletes the original. It then displays the copy and the batch file. To execute this .BAT file with your own parameters, type:

ASMFIL A:MYPROG B:YOURPROG

and press the **RETURN** key. The result is the same as if you had typed each of the following commands at your terminal.

```
REM This is ASMFIL.BAT
COPY A:MYPROG.ASM B:YOURPROG.ASM
DEL A:MYPROG.ASM
TYPE B:YOURPROG.ASM
TYPE ASMFIL.BAT
```

Pressing and holding the **SHIFT** key and pressing the **BRK/PAUS** key while in batch mode causes the following prompt to appear.

Terminate batch job (Y/N)?

If you respond by typing the letter **N**, the batch job will resume execution. If you respond by typing the letter **Y**, the remainder of the commands in the batch file are ignored and the system prompt appears.

AUTOEXEC.BAT File

When you boot your system, COMMAND.COM searches for the file AUTOEXEC.BAT on the drive you are booting from. If a file with that name exists on the diskette, then the batch facility automatically executes the commands contained in AUTOEXEC.BAT. This bypasses the execution of the TIME and DATE commands at start-up.

If COMMAND.COM does not find AUTOEXEC.BAT, then the normal MS-DOS prompt is displayed as described in Chapter 1, “System Start-up.”

You may define AUTOEXEC.BAT if you wish. This particular file is optional. As an example of an AUTOEXEC.BAT file, we repeat an example from the previous section:

```
1:  REM This is file AUTOEXEC.BAT
2:  PAUSE
3:  DIR
4:  CHKDSK
```

This set of commands first causes MS-DOS to return the remark, “This is file AUTOEXEC.BAT”. The execution of the batch file then pauses, and the remark “Strike a key when ready” is displayed. When you press a key, execution of the batch file is resumed. The computer then displays the directory listing and checks the diskette.

Command Directory

Check Disk (CHKDSK)	3-5
Allocation Error	3-6
Disk Not Initialized	3-6
Directory Error	3-7
Files Cross-Linked	3-7
Size Error	3-7
XXXXX Bytes Disk Space Freed	3-7
Configure (CONFIG)	3-8
COPY	3-12
Not Specified	3-14
Drive Destination Only	3-14
Filename Only	3-15
File Specification	3-15
COPY with Concatenation	3-16
Ambiguous Filenames	3-16
With Switches	3-18
DATE	3-20
Delete (DEL)	3-21
Directory (DIR)	3-22
Diskette Compare (DISKCOMP)	3-23
Diskette Copy (DISKCOPY)	3-26
ERASE	3-28
EXE to Binary (EXE2BIN)	3-29
FORMAT	3-31
PAUSE	3-33
Remark (REM)	3-34
Rename (REN)	3-34
System (SYS)	3-35
TIME	3-36
TYPE	3-38
MS-DOS COMMANDS	3-38

This chapter describes MS-DOS commands as they are used with the Texas Instruments Professional Computer. Commands are listed in alphabetical order—each with a description of its operation and a practical example of how it is used. At the end of this chapter is a table summarizing the MS-DOS commands and their use for quick and easy reference.

NOTE

Users of single-drive systems should refer to Appendix A for the additional procedures required for executing many of the commands described in this chapter.

The following notation is used in the descriptions in this chapter.

The **filespec** refers to an optional drive designation, followed by a filename, followed by a period and an optional three-letter filename extension. For example:

B:ABODE.BAS	refers to a diskette file on drive B
FILE.BAS	refers to a diskette file on the default drive
CON	refers to your display unit (console)
CON.BAS	same as above

The **filename** refers to any valid name for a diskette file, including an optional extension. A filename parameter does not refer to a device or to a diskette drive designation alone.

The **d:** refers to a diskette drive designation.

The **.ext** refers to a filename extension.

The following is a list of guidelines which apply to the use of all MS-DOS commands. For more detailed information on the different types of MS-DOS commands and how you use them refer to Chapter 2, “Working With MS-DOS.”

- Commands are usually followed by one or more parameters. Both the commands and their parameters can be entered in upper case or lower case or a combination of both.
- You must separate commands and parameters with a legal delimiter. The legal delimiters are a space, a comma, a semi-colon, or an equal sign. You can use any combination of delimiters within the same command.
- You can use wild-card characters and device filenames in a command parameter, but not in the command name.
- Commands start to execute whenever you press the **RETURN** key.
- You can stop execution of a command while it is running by pressing and holding the **SHIFT** key and then pressing the **BRK/PAUS** key. To suspend display of a command’s output, press the **BRK/PAUS** key. You can then press any key to resume display.
- The three parts of a filespec (d:filename.ext) must be separated by the colon (:) and the period (.) delimiters shown. MS-DOS does not accept any other delimiters in a file specification.
- When you create or rename files, they do not have to have an extension. You must, however, include the filename extension when you are referring to a file that has a filename extension.

CHECK DISK (CHKDSK)

Syntax: CHKDSK [d:]

Type: External

Function: The CHKDSK command is used to scan the directory of the designated drive and check it for errors.

Use: CHKDSK should be run occasionally on each diskette to verify that there are no errors in the directory structure. If any errors are found, the appropriate error messages are displayed and corrective action is attempted by the computer.

After the diskette has been checked, error messages (if any) and a status report are displayed. The status report tells you how much memory has been used and how much is still available. The following is a sample status report:

```
160256      bytes total disk space
  8192      bytes in 2 hidden files
 30720      bytes in 8 user files
  1024      bytes in bad sectors

120320      bytes available on disk
 65536      bytes total memory
 53152      bytes free
```

If an error is detected, one of the following error messages is returned:

- Allocation error for file: filename
- Disk not initialized
- Directory error file: filename
- Files cross-linked: filename and filename
- File size error for file: filename
- XXXXXX bytes of disk space freed

These error message are discussed individually below.

Allocation Error

The diskette has been damaged in some way. When the file-allocation table (FAT) stores a file, it separates the file into blocks of data which are stored in different locations on the diskette. This error tells you that one of the data blocks does not exist according to the FAT. The file is ended short of the bad block.

Disk Not Initialized

No directory or FAT was found. If files are on the diskette, but the diskette has been physically harmed you may still be able to recover the files. Try transferring the files from the damaged diskette to another formatted diskette using the COPY command.

Directory Error

No data blocks are allocated to the named file. The file is deleted.

Files Cross-Linked

This error is also indicative of diskette damage. The same data block has been allocated to two different files. MS-DOS doesn't do anything in this case. You can solve the problem by first using the COPY command to make copies of both files, then delete the originals. Review each file for validity. You may find it necessary to edit the files.

Size Error

The size of the file in a directory is different from its actual size. The size in the directory is automatically adjusted to indicate the actual size on the diskette. (The amount of useful data may be less than the size shown because the last data block may not be used fully.)

XXXXX Bytes Disk Space Freed

The amount of space on the diskette in bytes (represented by the letter X) shown as allocated, was actually not being used for anything. This diskette space is opened up to be used.

CONFIGURE (CONFIG)

Syntax: CONFIG P[rinter]=[P[arallel]][S[erial]][/N][/S]
 CONFIG S[erial]=port,[speed],[parity],[stopbits],
 [bustype][/N] [/S]

Type: External

Function: This command is used to modify the printer selection and serial printer configuration.

Use: To display your current system configuration enter the command:

CONFIG

The following message is returned.

```
Texas Instruments Professional Computer Configuration Utility version 1.10
```

```
Printer=Parallel  
Serial=P4_port, 9600_baud, Even_parity,  
7_databits, 1_stopbits, Scf_busy
```

The printer can be configured as either parallel or serial. Only the first letter of each parameter is required. For example, to designate a parallel configuration, enter:

CONFIG P=P

After each command is executed, a report of the current configuration is displayed. To suppress the report, you use a special letter called a switch. Switches are always preceded by a slash (/). If you don't want to display the report, put an /N switch at the end of the command. An /N switch at the end of the command causes the report to be suppressed.

If you designate a serial configuration, there are several possible values for each of the parameters. The values are listed in the following table with the original default values (from the factory) preceded by an asterisk.

Designating a serial configuration and changing the default parameters on your system requires two separate commands. The first command sets the serial configuration:

CONFIG P=S

or

CONFIG Printer=Serial

The second command sets the values for the parameters. To change the configuration of any device, you must type in the new configuration in the order shown in the command syntax. If you don't want to change a parameter from its current position, enter a comma for its position in the command. A comma must be entered for each position up to the last one changed.

```
Serial=*  P4_port,  110_baud,  Odd_parity,  *7_databits,  *1_stopbit,  N0_busy
Y        P3        150        *Even        8        2        Xon_Xoff
ff       P2        300        No           8        2        Dsr_busy
SY       P1        600                8        2        scf_busy
SY                1200
                2400
                4800
                *9600
```

For example assume you have already designated a serial configuration for your system. To change the serial port to 1200 baud and stopbits to 2, but leave the other parameters unchanged, you can use the following command.

CONFIG Serial= , 1200__baud, , ,2__stopbits

Note that lowercase letters and underscores shown in the command syntax do not have to be typed and can, in fact, be any alphabetic character. We have included them merely to clarify our examples. The above example could also have been entered like this:

CONFIG S=,1200,,,2

To save the current configuration on the diskette so that it will not change each time the system is rebooted, put a /S switch at the end of the command. The following are some examples using the /S switch and the /N switch.

CONFIG /S	(Reports configuration and saves it)
CONFIG Printer=Serial /s	(Sets printer to serial and saves it)
config s=p1,9600,n,8,1,x/s	(Sets serial configuration and saves it)
CONFIG p=p/n	(Changes to parallel port without report)

If you make an error in entering the command, the following message is printed followed by the command up to the point where the error was found.

Entry error at or before the last character of the next line.

For example entering the following command with no entry in the parity position:

```
CONFIG SERIAL=P4,1200,8,1,N
```

would result in the message:

```
Entry error at or before the last character of  
the next line.
```

```
SERIAL = P4, 1200, 8
```

COPY

Syntax: COPY [/A][/B] filespec[/A][/B] [d:][filename[.ext]][/A][/B][/V]
or
[COPY [/A][/B] filespec [/A][/B] [+ filespec[/A][/B]
[d:][filename[.ext]][/A][/B][/V]

Type: Internal

Function: The COPY command can be used to copy the contents of one or more files to another diskette with either the same filename or a name you specify. You can also copy one or more files to the same diskette (in this case the copies must have different names). With the COPY command you can also combine files in a process referred to as concatenation described in the next section.

Use: The first filespec parameter is referred to as the source file. The second, d: filename.ext parameter, is the destination file. This is the file in which the copy will be placed. Note that if the destination file exists, its contents are replaced by the contents of the source file.

The /V switch verifies that the original and its copy are identical. If the copy is verified (which is usually the case), no message is returned. If an error is detected in the copy, a disk error message is displayed. You then have the option of telling the computer to Abort, Ignore, or Retry the copy. Although this switch causes the COPY command to run more slowly, it's still a good idea to use it when you are making copies of critical data.

The /A and /B switches tell the COPY command how much of the file is to be processed. Each switch applies to the filespec preceding it and to all remaining filespecs on the command line until another /A or /B is found. These switches have varying effects depending on the parameter to which they refer.

The /A switch has the following effect:

- When used with the *source* filespec, it causes the file to be treated as a text or ASCII file. A CTRL-Z or 1A hexadecimal in the file is interpreted as the end-of-file marker. The file is not copied beyond the end-of-file marker.
- When used with the *destination* filespec, it causes the physical end of file (as seen in the DIR command) to be interpreted by the COPY command as the end-of-file. The entire file is copied.

The /B switch has the following effect:

- When used with the *source* filespec, it causes a CTRL-Z character to be added as the last character of the file.
- When used with the *destination* filespec, no CTRL-Z character is added at the end of the file.

The default values are /B when you are copying without concatenation and /A when concatenation is being performed.

The destination filespec may take one of four forms:

- Not specified
- Drive designation only (d:)
- Filename only
- Complete filespec

Not Specified

The source file is copied to the default drive and given the same name as the source file. If the destination file exists, its contents are replaced by the contents of the source file. For example the command

COPY A: MEMOS.TXT

copies the source file MEMO.TXT from the diskette in drive A to the default drive and gives it the name MEMO.TXT. If drive A is the default drive, then the following message is returned.

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

Drive Designation Only

The source file is copied onto the designated drive and is given the same name as the source file. For example the command

COPY MEMOS.TXT B:

copies the source file, MEMOS.TXT, which is located on the default drive, onto the diskette in drive B. If the default drive is drive B, MS-DOS would return the error message shown in the previous example.

Filename Only

The source file is copied to the drive designated in the second filespec and is given the filename designated in the second filespec. If no drive is specified in the second filespec, the default drive is assumed. For example the command

COPY MEMOS.TXT MEMOS.OLD

creates a copy of the source file, MEMOS.TXT, and gives it the name MEMOS.OLD. (The file MEMOS.OLD is created if it does not already exist.) Note that in the above example, both the source file and the destination file reside on the diskette in the default drive.

File Specification

The source file is copied onto the diskette in the drive designated in the second filespec and is given the filename designated by the second filespec. For example the following commands

COPY MEMOS.TXT B:MEMOS.NEW

COPY A:MEMOS.TXT B:MEMOS.NEW

both copy the source file MEMOS.TXT from the diskette in drive A (assuming drive A is the default drive) onto the diskette in drive B and give the copy the name MEMOS.NEW.

COPY WITH CONCATENATION

The COPY command also allows file concatenation while copying. File concatenation means that the contents of more than one source file can be placed in a larger destination file. Concatenation is invoked by placing a plus sign (+) between multiple source files.

For example, to combine the files MEMOS.TXT and MEMOS.OLD in another file named MEMOS.NEW, you could use the following expression.

COPY MEMOS.TXT + MEMOS.OLD MEMOS.NEW

Concatenation is usually carried out in the default mode, meaning a CTRL-Z or 1A hexadecimal in the file is interpreted as the end-of-file marker. To combine binary files the /B switch is used. This forces MS-DOS to look for the physical end of file (as seen in the DIR command) during the copy operation. The following example combines binary files.

COPY /B A.COM + B.COM

Unless you work with both binary and text files, further explanation of the use of switches in the concatenation process is not relevant. For those of you who are interested in knowing more about this process, refer to this section at the end of this command description entitled “With Switches.”

Ambiguous Filenames

By ambiguous filenames we mean simply filenames which use an asterisk (*) or a question mark (?) in their specification. MS-DOS allows you to concatenate ambiguous filenames, but you should exercise caution in doing so. If you want to be sure of exactly what files you are combining, use a DIR command to display the filenames before you combine them.

To combine several files specified with an ambiguous name into one file, use a command such as,

COPY *.LST + COMBIN.PRN

All files matching *.LST are combined into one file called COMBIN.PRN. Several individual concatenations can be done with the COPY command.

COPY *.LST + *.REF *.PRN

In this example, each file found matching *.LST is combined with the corresponding .REF file. The result is given the same filename and the extension .PRN. Thus, FILE1.LST is combined with FILE1.REF to form FILE1.PRN, and XYZ.LST is combined with XYZ.REF to form XYZ.PRN, and so on.

Now if you wanted to take all the files matching *.LST, combine them with all the files matching *.REF, and put them in one file called COMBIN.PRN, you could do so with the following COPY command.

COPY *.LST + *.REF COMBIN.PRN

It is easy to enter a concatenation COPY command where one of the source files is the same as the destination, yet this often cannot be detected. For example, the following command produces an error message if ALL.LST already exists.

COPY *.LST ALL.LST

This is not detected, however, until ALL.LST is ready to be appended. At this point ALL.LST could already have been destroyed.

The COPY command, however, has a utility to take care of this problem. As each input file is found, its name is compared with the destination filename. If the two names are the same, that one input file is skipped, and the following message is displayed:

Content of destination lost before copy

Further concatenation proceeds normally. This allows “summing” files, with a command like

COPY ALL.LST + *.LST

This command appends all *.LST files, except ALL.LST itself, to ALL.LST. The error message is suppressed in this case, since this is produced by a true physical append to ALL.LST.

With Switches

As you may recall from our earlier discussion, we outlined briefly the use of the /A and /B switches. The use of these switches becomes slightly more complex when concatenation is involved. The concatenation operation is normally carried out in text or ASCII mode, meaning a <CTRL-Z> (1A hexadecimal) in the file is interpreted as the end-of-file mark. To combine binary files, this interpretation of the end-of-file may be overridden with the /B switch.

ASCII and binary files may be arbitrarily combined by using a /B switch on binary files and a /A switch on ASCII files. A switch (/A or /B) takes effect on the file it follows. The switch applies to all subsequent files until another switch is found.

An /A or /B switch on the destination file determines whether or not a <CTRL-Z> is placed at the end of the file. Source files read while /A is in effect have <CTRL-Z> stripped off. If /A is in effect when the file is written, a single <CTRL-Z> will be put back. Thus, an additional <CTRL-Z> would be added with a command such as:

COPY A.ASM/B B.ASM/A

This occurs because the /B switch on the first file prevents the <CTRL-Z> from being stripped, and the /A switch on the second file puts one on. The primary practical application may be the reverse, where a <CTRL-Z> is stripped from the file. For example,

COPY PROG.COM/B + ERRS.TXT/A NEWPROG.COM /B

It is assumed here that ERRS.TXT was generated by an editor, but is actually considered constant data (error messages) by the program to which it is being appended. Since the result is a .COM file, a <CTRL-Z> at the end is not needed.

Even when not concatenating files, the /A and /B switches are still processed. By using the /A switch, the destination file may be truncated at the first end-of-file mark. In the example:

COPY A.TXT/A B.TXT

B.TXT may be shorter than A.TXT if A.TXT contains an embedded <CTRL-Z>. B.TXT will have exactly one <CTRL-Z>, the last character of the file.

DATE

DATE

Syntax: DATE [mm-dd-yy]

Type: Internal

Function: This command allows you to display and set the date. It is useful to be able to show the current date on your display, but it is more important for MS-DOS to have a correct date entered in the system. Whenever you update a file, MS-DOS makes a note of the date and time of the revision in the directory. This information does not do you any good unless the date is accurate.

Use: If you want to display the current date, type:

DATE

The following message is displayed.

```
Current date is mm-dd-yy
Enter new date:[]
```

where the letters mm-dd-yy stand for the current month, day, and year.

If you don't want to change the date shown, simply press the **RETURN** key. You may change the date by entering the new month, day, and year values as shown:

```
Enter new date: 10-25-82
```

Notice that the new date must be specified with numbers. The numbers that MS-DOS accepts are listed below.

```
mm = 1-12
dd = 1-31
yy = 80-99 or 1980-1999
```


You can use either hypens (-) or slashes (/) to separate the different parts to the date. MS-DOS changes months and years automatically. It even handles leap years.

You can also specify a new date when you enter the command. For example if you wanted to tell the computer it was August 3, 1982, you would enter:

DATE 8-3-82

In this case no message is returned.

If something happens that you enter an unacceptable character in the date, MS-DOS returns the following message.

**Invalid date
Enter new date:[]**

DELETE (DEL)

Syntax: DEL filespec

Type: Internal

Function: This command allows you to delete all the files with the specified filespec. You should always check a file before you delete it. There is no way to recover a deleted file. Exercise great caution when you use wild-card characters with this command. When using wild-card characters with the DEL command, it's a good idea to first execute a DIR command with the wild-card filespecs to see exactly what will be deleted.

DIR

Use: If you type *.* as a filespec, MS-DOS returns the message:

Are you sure?

If you type either an uppercase or a lowercase **Y** as your response, then all the files are deleted as requested.

ERASE is a synonym for this command.

DIRECTORY (DIR)

Syntax: DIR [filespec] [/P][/W]

Type: Internal

Function: This command enables you to list all directory entries or only those for specific files. It also displays the file sizes in bytes and the time and date of the last updates.

Use: The DIR command allows you to list all the directory entries on a designated drive or only those of specified files. To list all the entries, type:

DIR
or
DIR d:

The second example lists the directory entries on a drive other than the default drive.

To list the directory entries of specific files, use the same command followed by the specified filespec. The command

DIR d:filename.ext

lists all the entries from the designated drive that have the specified filespec. Files are listed with their size in bytes and the time and date of their last modification.

The filespec may include wild-card characters to show groups of files. Refer to Chapter 2 for more information on wild-card characters.

The /P switch selects page mode. The /P switch causes the directory listing to pause after the display is filled. To resume the directory listing press any key.

The /W switch selects wide display. The /W switch causes only filenames to be displayed, without information about the file size and the date and time of the last update. Filenames are displayed five per line.

The MS-DOS system files IO.SYS and MSDOS.SYS are not listed, if present.

DISKETTE COMPARE (DISKCOMP)

Syntax: DISKCOMP [d:] [d:][/1]

Type: External

Function: This command compares the contents of one diskette to that of another. DISKCOMP is often used after a DISKCOPY command to verify that the original diskette and the copy are identical.

Use: The first two parameters in the DISKCOMP command refer to the drives that contain the diskettes you want to compare. You can specify the same drive for both parameters, or the drive specifications can be different.

A /1 switch in the filespec compares only the first side of the diskettes, even if the diskettes and drives are dual sided.

If you specify the same drive for both parameters, a single-drive comparison is performed on the specified drive. If you omit both parameters, a single-drive comparison is performed on the default drive. DISKCOMP prompts you when to insert and when to remove the diskettes and waits for you to press any key before continuing.

If you omit the second parameter, the default drive is assumed to be the second drive. If you omit the second parameter and you specify the default drive in the first parameter, a single-drive comparison is performed.

On a single-drive system, all prompts are for drive A, regardless of the designated drive.

All of the diskette tracks are compared on a track-for-track basis. If the tracks are not equal, DISKCOMP issues a message indicating the track number (0 through 79) and side (0 or 1) of the track that doesn't match.

If diskettes are identical, the following message is returned.

Diskettes compare OK

This message will probably not appear if you are comparing a backup diskette made with the COPY command to its original. The COPY command produces a copy with the same information as the original diskette, but it may be stored in different locations. Therefore the two diskettes may not be identical. Chapter 6 describes the FILCOM command which you can use to compare the diskettes on a file-by-file basis.

After the diskette comparison completes, the following message is displayed.

Compare more diskettes (Y/N)?

If you type **Y**, DISKCOMP prompts you to insert the proper diskette, then executes another comparison on the drives you specified originally.

Typing **N** ends the command.

There are a few things to keep in mind while using the DISKCOMP command.

- The DISKCOMP command automatically determines the number of sides to be compared based on the diskette that is read first (specified by the first drive parameter). If the first diskette can be read only on one side, or if a /1 switch is used, only the first side is read from both diskettes.
- If the first drive and diskette are dual sided, a two-side comparison is performed. In this case, if either the second diskette or drive is single sided, an error message is returned.
- If a diskette error occurs while the diskette is being read, an error message is produced which tells the location (track and side) of the error. DISKCOMP then continues to compare the rest of the diskette.

DISKETTE COPY (DISKCOPY)

Syntax: DISKCOPY [d:] [d:][/1][/F][/V]

Type: External

Function: This command copies the contents of one diskette onto another. Unlike the COPY command, which causes information to be copied file by file, DISKCOPY causes the original diskette to be copied track by track in its entirety.

Use: The first parameter specified in the filespec refers to the original or source diskette drive. The second parameter specified refers to the drive that contains the copy, the destination drive. You may specify both parameters with the same drive designations, or the drive designations may be different.

A /1 switch in the filespec causes only the first side of the diskette to be copied, regardless of the diskette or drive type.

A /F switch in the filespec causes your destination diskette to be formatted before the copy operation is performed. This switch functions similarly to the MS-DOS FORMAT command discussed later in this chapter.

A /V switch in the filespec causes the source diskette to be reread and a full source-to-destination comparison to be performed. This procedure verifies the accuracy of your copy. Without the /V switch, DISKCOPY would simply compare the destination diskette with the information that is still in main memory. The /V switch is similar in function to the DISKCOMP command. Although including this switch causes the DISKCOPY operation to be slower, it is recommended that you always verify your copies.

If you specify the same drive for both parameters, a single-drive copy operation is performed on the designated drive. If you omit both parameters, a single-drive copy operation is performed on the default drive. In either case, DISKCOPY prompts you to insert and remove diskettes and waits for you to press any key before continuing.

If you omit the second parameter, the default drive is assumed to be the destination drive. If you omit the second parameter and specify the default drive in the first parameter, a single-drive copy operation is performed.

On a single-drive system, all prompts are for drive A, regardless of the designated drive.

After copying has been completed, the following message is displayed.

Copy another (Y/N)?

If you type **Y**, DISKCOPY prompts you to insert the proper diskette, then executes another copy operation on the same drives you specified originally.

Typing **N** ends the command.

There are a few things to keep in mind when using the DISKCOPY command.

- DISKCOPY automatically determines the number of sides to be copied based on the source drive and diskette. If either the diskette or the drive are single sided, only one side can be copied. If both the drive and the diskette are dual sided, both sides are copied (unless the filespec contains a /1).

- If both the source drive and diskette are dual sided and the destination drive is single sided, an error message will result.
- If an error on either diskette is encountered during the operation DISKCOPY indicates the location of the error (drive, track, and side) and continues copying. If this happens the destination diskette may or may not be usable.
- Information on diskettes that have undergone a lot of creation and deletion activity may not be allocated sequentially. This is referred to as “fragmentation.” A fragmented diskette may not perform as well as its newer counterparts. To alleviate this problem, you should use the COPY command instead of DISKCOPY. This copies the diskette file by file. For example, to copy all the files from the diskette in drive A to the diskette in drive B, you would use the following command.

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

ERASE

Syntax:	ERASE filespec
Type:	Internal
Function:	This command allows you to delete all the files with the specified filespec.
Use	This command functions identically to the DEL command discussed earlier in this chapter.

EXE TO BINARY (EXE2BIN)

Syntax: EXE2BIN filespec [d:][filename][.ext]

Type: External

Function: This command converts files from .EXE format to binary format. Unless you are an experienced programmer, this command will not be particularly useful to you.

Use: The first parameter is the input file. If no extension is given on the first filespec, it defaults to .EXE.

The second parameter is the output file. If no drive is designated in the second parameter, the drive of the input file is used. If no filename is given in the second parameter, the filename of the input file is used. If no extension is given on the second parameter, .BIN is used.

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

Two kinds of conversion are possible. The initial CS:P specifies the conversion. The conversions are:

1. If CS:IP is not specified, a pure binary conversion is assumed. If segment fix-ups are necessary, the following prompt appears:

Fix-up needed - base segment (hex):

By typing an acceptable hexadecimal number and pressing the **RETURN** key, execution continues.

-
2. If CS:IP is specified as 100H, then it is assumed the file is to be run as a .COM file ORGed at 100H, and the first 100H of the file is to be deleted. No segment fix-ups are to be allowed, as .COM files must be segment relocatable.

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

File cannot be converted.

Note that to produce standard .COM files with the MACRO-86 assembler, one must both ORG the file at 100H and specify the first location as the start address (this is done in the END statement).

For example:

```
                ORG      100H

START:
    .
    .
    .
END            START
```


FORMAT

Syntax: `FORMAT d: [/S][/1]`

Type: External

Function: This command formats the diskette in the designated drive so that it will accept MS-DOS files.

Use: The `FORMAT` command initializes the directory and file allocation tables. You must format all new diskettes—either with the `FORMAT` or the `DISKCOPY` command—before they can be used by MS-DOS. The portion of the MS-DOS diskette which contains start-up information (the reserved sectors) is automatically copied onto the diskette you are initializing. This occurs whether or not the `/S` switch is given.

The `/S` switch is an optional appendage to the `FORMAT` command. If the `/S` switch is present, `FORMAT` copies the MS-DOS system files—`IO.SYS`, `MSDOS.SYS`, and `COMMAND.COM`—from the diskette in the default drive to the newly formatted diskette. Since `IO.SYS` and `MSDOS.SYS` are hidden files, a `DIR` command performed on the newly formatted diskette will only list `COMMAND.COM`.

If you include a /1 on the command line, the destination diskette is formatted for single-sided use, regardless of the drive type. If you do not use the /1 switch, the diskette is formatted according to the destination drive type. If the drive is single sided, the diskette is formatted for single-sided use and can be used in either type of drive. If the destination drive is dual sided and you do not use a /1 switch, the diskette is formatted for dual-sided use, and it is not usable in a single-sided drive.

The FORMAT command also produces a status report which tells you how much space on the diskette is currently being used and how much is available for you to use.

For example to format a new diskette in drive A and copy the MS-DOS system files onto it, you could use the following command.

FORMAT A:/S

The following message is returned.

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

When the format operation is complete, FORMAT displays the status report.

System transferred

160256	bytes total disk space
15872	bytes used by system
1024	bytes in bad sectors
143360	bytes available on disk
Format another (Y N)?[]	

To format another diskette, type a **Y**. To end the FORMAT program, type an **N**.

PAUSE

Syntax: PAUSE [comment]

Type: Internal

Function: This command suspends the execution of a batch file. If you don't intend to use batch files in your work, then this command will not be particularly useful to you.

Use: During the execution of a batch file, you may need to change diskettes or to perform some other action between the execution of batch commands. The PAUSE command allows you to suspend execution. To resume execution, press any key except the **BRK/PAUS** key.

When MS-DOS encounters PAUSE, it displays:

Strike a key when ready . . .

Pressing any key except the **BRK/PAUS** key resumes execution of the batch file. If you press and hold the **SHIFT** key and then press the **BRK/PAUS** key, the following prompt is displayed:

Terminate batch job (Y/N)?

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

The comment portion of this command is optional. If you decide to use a comment, it must be entered on the same line as PAUSE.

Comments are useful if you want to prompt the user of the batch file with a message when the batch file has paused. For example, you may want to change diskettes in one of the drives. An optional prompt message may be given in such cases. The comment prompt is displayed before the “Strike a key” message.

REMARK (REM)

Syntax: REM [comment]

Type: Internal

Function: The REM command is used with the execution of a batch file. This command displays your entered remark when REM is encountered during the execution of a batch file. If you don't intend to use batch files in your work, this command will not be particularly useful to you.

RENAME (REN)

Syntax: REN filespec filename

Type: Internal

Function: This command changes the name of the first parameter (filespec) to the second parameter (filename).

Use: If the first filespec is not on the default drive, a drive designation must be specified. Any drive designation for the second filename is ignored. The renamed file must be on the same diskette as the original.

The wild-card characters, question mark (?) and asterisk (*), may be used in either parameter. If you use a wild-card character in one parameter you must use it in the other parameter. Use of a wild-card character creates a one-to-one correspondence between the files designated by the filespec and the files designated by the filename.

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 following example the file ABODE on drive B is renamed ADOBE:

REN B:ABODE ?D?B?

The file remains on drive B.

Note that RENAME is a synonym for the REN command.

SYSTEM (SYS)

Syntax: SYS d:

Type: External

Function: This command transfers the MS-DOS system files from the diskette in the default drive to the drive specified by d:.

Use: SYS is normally used to update a system. An entry for d: is required.

TIME

The files transferred are copied in the following order.

IO.SYS
MSDOS.SYS

Note that COMMAND.COM is not transferred. Also note that IO.SYS and MSDOS.SYS are both hidden files. They do not appear when the DIR command is executed.

TIME

Syntax: TIME [hh:mm:ss.hh]

Type: Internal

Function: This command allows you to display and set the time.

Use: If you want to display the current time, simply type

TIME

MS-DOS returns the message

```
Current time is hh:mm:ss.hh  
Enter new time:[]
```

If you don't want to change the date displayed, simply press the **RETURN** key. You may change the time by entering the new hour, minute, and second (optional) and hundredths (optional) values as shown:

```
Enter new time: 8:20:00
```

Notice that the new time must be specified with numbers. The numbers MS-DOS accepts are listed below.

hh = 00–24
mm = 00–59
ss = 00–59
hh = 00–99

The hour, minute, and second values must be separated by colons and the hundredths by a period (if entered).

You can also specify a new time when you enter the command. For example if you wanted to tell the computer it is 2:30 P.M., you would enter

TIME 14:30:00

In this case, MS-DOS does not return a message.

If you enter an unacceptable character in the time, MS-DOS returns the message:

**Invalid time
Enter new time:[]**

MS-DOS then waits for the entry of an acceptable time.

TYPE

TYPE

- Syntax: TYPE filespec
- Type: Internal
- Function: This command shows the contents of a file on your display.
- Use: You can use this command to examine a file without modifying it. You use DIR to find the name of a file and EDLIN to alter the contents of a file. The TYPE command causes tabs to be expanded to spaces consistent with tab stops every eighth column.

Note that displaying binary files causes control characters to be sent to your computer, including bells, formfeeds, and escape sequences.

To print the contents of the file as they are displayed, press the **PRNT** key after you have entered the TYPE command, but before you press the **RETURN** key.

MS-DOS COMMANDS

The following table is a summary of the MS-DOS commands; each with its type, syntax, and function.

Command	Type	Syntax	Function
CHKDSK	E	CHKDSK [d:]	Scan the directory of the designated drive and check it for errors.
CONFIG	E	CONFIG P[rinter]=[P[arallel]] [S[erial]][/N][/S] or CONFIG S[erial]=port,[speed], [parity],[stopbits], [busytype] [/N][/S]	Modify printer selection and serial printer configuration.
COPY	I	COPY [/A][/B]filespec[/A][/B] [d:][filename[.ext]] [/A][/B][/V] or COPY [/A][/B]filespec[/A][/B] [+ filespec[/A][/B][d:] [filename[.ext]][/A][/B][/V]	Copy the contents of the first file to the second file.

E represents external commands.

I represents internal commands.

Command	Type	Syntax	Function
DATE	I	DATE [mm-dd-yy]	Display and set the date.
DEL	I	DEL filespec	Delete the file(s).
DIR	I	DIR [d:][filespec][P][W]	List the specified filename(s).
DISKCOMP	E	DISKCOMP [d:] [d:][/1]	Compare the contents of the first diskette to the second diskette.
DISKCOPY	E	DISKCOPY [d:] [d:][/1][F][V]	Copy the contents of the first diskette to the second diskette.
ERASE	I	ERASE filespec	Synonym of DEL.
EXE2BIN	E	EXE2BIN filespec [d:] [filename[.ext]]	Convert files from .EXE format to format
FORMAT	E	FORMAT d: [/S]	Format the diskette.
PAUSE	I	PAUSE [comment]	Suspend execution of the batch file.

E represents external commands.

I represents internal commands.

Command	Type	Syntax	Function
REM	I	REM [comment]	Display remark in batch file.
RENAME	I	REN[AME] filespec filename	Change the name of the first filespec to second filename.
SYS	E	SYS d:	Transfer MS-DOS system files to designated drive.
TIME	I	TIME [hh:mm:ss.hh]	Display and set the time.
TYPE	I	TYPE filespec	Display the contents of a file.

E represents external commands.

I represents internal commands.

4

Line Editor

Terms	4-3
Starting EDLIN	4-4
Intraline Commands	4-9
RIGHT ARROW	4-10
F2	4-11
F3	4-12
DEL (Delete)	4-13
F4	4-14
DOWN ARROW	4-15
INS (Insert Mode)	4-16
F5	4-17
Interline Commands	4-19
Parameters	4-19
Edit	4-21
Append	4-23
Delete	4-23
End	4-27
Insert	4-28
List	4-33
Quit	4-36
Replace	4-38
Search	4-43
Write	4-46
Errors When Invoking EDLIN	4-47
Errors While Editing	4-48

The MS-DOS line editor is called EDLIN. You use EDLIN to create and edit your text files. EDLIN works on a line-by-line basis through the use of special-editing keys and a few simple commands. These special keys are identical to the command-line editing keys discussed in Chapter 2.

EDLIN allows you to create new files and change your existing files by editing, inserting, deleting, and otherwise manipulating lines of text.

This chapter is a directory with the special editing keys and EDLIN commands arranged so you can refer to them easily. We suggest that you read through this chapter with your Texas Instruments Professional Computer in front of you and practice each key and command as you come to it. As soon as you gain familiarity with EDLIN, you'll find you only need to use this chapter for reference.

TERMS

EDLIN manipulates lines of text. Each line may be up to 255 characters long. The last character is the end-of-line character, the carriage return. Although line numbers are not actually present in the saved text, EDLIN numbers the lines whenever a file is displayed.

When a file is created or edited, line numbers begin at one and are incremented by one through the end of the file. When new lines are inserted between existing lines, all line numbers following the inserted text are automatically incremented by the number of lines inserted. When lines are deleted between existing lines, all line numbers following the deleted text are decremented automatically by the number of lines deleted. Line numbers always range consecutively from one to the number of the last line in the file.

EDLIN commands belong to two types: intraline and interline. Intra-line commands perform editing functions within a single line. These are the special editing keys and control characters discussed in Chapter 2. They function identically whether you are editing a line of text or the command line. This chapter describes in more detail what these keys do and how you should use them.

Interline commands perform editing functions on entire lines. You use these commands much like you use the MS-DOS commands described in Chapter 3. The main difference between interline commands and MS-DOS commands is that while MS-DOS commands function any time MS-DOS is loaded, interline commands function only after you have entered the EDLIN program. This simple procedure is referred to as “invoking” EDLIN and is described in the next section.

STARTING EDLIN

To work with a file in EDLIN, enter:

EDLIN < filespec >

where filespec represents the name of the file you wish to use. If the file you specify does not exist, EDLIN creates the file and returns the message:

NEW FILE

The asterisk prompt (*) is then displayed, indicating that the editing session may begin.

NOTE

When creating a new file, be sure to specify which drive the file should be saved on. The command to end the editing session and save the file does not allow parameters. Therefore, if the drive is not designated during EDLIN invocation, the file is saved on the default drive.

Suppose you want to create a file named PRACTICE and save it on drive A. Type the command:

EDLIN A:PRACTICE

and press the **RETURN** key. MS-DOS returns the following prompt indicating that the file has been created.

NEW FILE
***[]**

To begin inserting lines into the file, type the letter **I** then press the **RETURN** key. MS-DOS returns the line number followed by the asterisk prompt. You can now insert lines into your file.

New file
***I**
1:*[]

When you are finished inserting lines into the file, you can do one of two things to return to the EDLIN command prompt. You can:

1. Press and hold the **SHIFT** key and then press the **BRK/PAUS** key. (This is represented as <SHIFT-BRK>.) The line you are currently editing is aborted, and the EDLIN command prompt is displayed. Pressing and holding the **CTRL** key and then typing **C** has the same effect as pressing and holding the **SHIFT** key and then pressing the **BRK/PAUS** key.
2. Press and hold the **CTRL** key, then type **Z** at the start of a new line, and then press the **RETURN** key. The EDLIN command prompt is displayed.

Insert the following lines into your file PRACTICE

```
1: *This is a sample file
2: *used to demonstrate
3: *how to create a file in EDLIN.[]
```

with the cursor at the end of line 3. Pressing and holding the **SHIFT** key while pressing the **BRK/PAUS** key has the following effect.

```
1: *This is a sample file
2: *used to demonstrate
3: *how to create a file in EDLIN.
*[]
```

Now type the letter **L** and press the **RETURN** key to list the file.

```
*L
1:  This is a sample file
2:  used to demonstrate
*[]
```

Notice that line 3 has been aborted. Now type the letter **I** and press the **RETURN** key. You can now resume inserting text. Retype line 3 and press the **RETURN** key.

```
*I
3: *how to create a file in EDLIN
4: *[]
```

Now press and hold the **CTRL** key and type a **Z**, and then press the **RETURN** key. Your display should look like this.

```
3:*how to create a file in EDLIN
4:* ^Z
*[]
```

Type the letter **L** and press the **RETURN** key to list the file.

```
*L

1:  This is a sample file
2:  used to demonstrate
3:  how to create a file in EDLIN.
*[]
```

To quit editing the file and discard all your changes, type the letter **Q** and press the **RETURN** key.

```
*Q
Abort edit Y/N[]
```

If the file you are editing has been created in a session prior to the current one, typing the letter **Y** would cause editing session to stop and all your changes from that section to be discarded. In our case typing the letter **Y** would cause the file PRACTICE to be deleted since it is a new file and has not been saved on the diskette.

To return to the file, type the letter **N** and press the **RETURN** key. To save the file and keep your modifications, type the letter **E** and press the **RETURN** key. The file PRACTICE is now saved on drive A.

```
*Q
Abort edit Y/N N
*E
```

```
A: [ ]
```

If the file you specify exists, EDLIN loads the file into memory. If the entire file is loaded, EDLIN returns the message “End of input file,” and an asterisk (*) prompt. If the file is too large to fit in memory all at once, EDLIN loads three-fourths of the file, or as much as available memory can hold. In this case only the asterisk (*) prompt is displayed.

You may then edit the existing file. When you want to edit the part of a file that is not in memory, you must first transfer some of the file that is in memory to the diskette. You can then append lines into memory.

To transfer or “write” lines from memory to the diskette you use the following command

```
[n]W
```

where n is the number of lines you want written. The command to append lines into memory is

```
[n]A
```

where n is the number of lines to be appended.

When the editing session ends, save the file on the same drive where it was found by typing:

E

Remember, at this point, that if you created a file and did not specify a drive in the filespec, the file will be saved on the default drive.

The rest of this chapter is divided into two major sections—intraline commands and interline commands. The best way to learn to use EDLIN efficiently is to practice using the commands. Work through these sections until you are comfortable with the commands and special editing keys. It's much easier to work when you don't have to refer to this manual every time you need to accomplish a simple task.

INTRALINE COMMANDS

As we stated earlier, the intraline commands include the special editing functions and the control-character functions. This section discusses only the special editing functions. See Chapter 2, "Working With MS-DOS," for more information on the control-character functions. The following table lists the special editing keys with a brief summary of their functions.

Key	Function
F2	Copy all characters up to the character specified from the template to the current line.
F3	Copy all remaining characters in the template to the current line.
F4	Skip over (do not copy) the characters in the template up to the character specified.
F5	Output an @ (at sign) and make the new line the new template

Key	Function
RIGHT ARROW	Copy one character from the template to the current line.
LEFT ARROW	Backspace.
DOWN ARROW	Stop editing current line without changing the template.
INS	Enter insert mode.
DEL	Skip over (do not copy) a character in the template.

RIGHT ARROW

Function: Copy one character from the template to the input buffer.

Use: Pressing the **RIGHT ARROW** key copies one character from the template to the input buffer. When the **RIGHT ARROW** key is pressed, one character is inserted in the buffer and insert mode is automatically turned off if it was on. Use the **RIGHT ARROW** key to advance the cursor one column across the line.

For example assume the display shows:

```
1: *This is a sample file.  
2: *[]
```

with the cursor positioned at the beginning of the second line. Pressing the **RIGHT ARROW** key copies the first character (T) to the second of the two lines displayed:

```
1:*This is a sample file
2:*T[]
< RIGHT ARROW>
```

Each time the **RIGHT ARROW** key is pressed, one more character appears:

```
< RIGHT ARROW> 1:*Th[]
< RIGHT ARROW> 2:*Thi[]
< RIGHT ARROW> 2:*This[]
```

F2

Function: Copy multiple characters up to a given character from the template to the new line.

Use: Pressing the **F2** key copies all characters up to a given character from the template to the input buffer. The given character is the next character typed after pressing the **F2** key and is not copied or shown on the display.

If the template does not contain the specified character, nothing is copied. Pressing the **F2** key also automatically turns off the insert mode if it is on.

For example assume the display shows:

```
1:*This is a sample file.
2:*[]
```

with the cursor positioned at the beginning of the second line. Pressing the **F2** key then typing the letter **p** copies all characters up to the first occurrence of the letter p.

```
1: *This is a sample file  
< F2> p 2: *This is a sam[]
```

F3

Function: Copy the template to the input buffer.

Use: Pressing **F3** copies all remaining characters from the template to the input buffer. Regardless of the cursor position at the time the **F3** key is pressed, the rest of the line appears, and the cursor is positioned after the last character on the line.

For example assume the display shows:

```
1: *This is a sample file.  
2: *[]
```

Pressing the **F3** key copies all characters from the template (shown in the upper line displayed) to the line with the cursor (the lower line displayed):

```
1: *This is a sample file  
< F3> 2: *This is a sample file.[]
```

Also, insert mode is automatically turned off if it is on.

DEL (Delete)

Function: Skip over one character in the template

Use: Pressing the **DEL** key skips over one character in the template. Each time you press the **DEL** key, one character is deleted (not copied) from the template.

The action of the **DEL** key is similar to the **RIGHT ARROW** key, except that the **DEL** key skips a character in the template instead of copying it to the input buffer.

For example assume the display shows:

```
1:*This is a sample file.  
2:*[]
```

Pressing the **DEL** key skips over the first character (T).

```
1:*This is a sample file  
< DEL> 2:*[]
```

The cursor does not move, only the template is affected. To see how much of the line has been skipped over, press the **F3** key. This copies the remainder of the template onto the input buffer and moves the cursor beyond the last character of the line.

```
1:*This is a sample file.  
< DEL> 2:*[]  
< F3> 2:*his is a sample file.[]
```

F4

Function: Skip over multiple characters in the template

Use: Pressing the **F4** key followed by a given character skips over all characters in the template up to the given character. The given character is the next character typed after you press the **F4** key. This character is not copied and is not shown on your display. If the template does not contain the specified character, nothing is skipped over.

The action of the **F4** key is similar to the **F2** key, except that the **F4** key skips over characters in the template instead of copying them to the input buffer.

For example assume the display shows:

```
1:*This is a sample file.  
2:*[]
```

Pressing the **F4** key followed by typing the letter **p** skips over (deletes) all characters in the template up to the the first occurrence of the letter p.

```
1:*This is a sample file  
< F4> p 2:*[]
```

The cursor does not move. To see how much of the line has been skipped over, press the **F3** key to copy the the remainder of the template. This moves the cursor beyond the last character of the line:

```
1:*This is a sample file:  
< F4> p 2:*[]  
< F3> 2:*ple file.[]
```

DOWN ARROW

Function: Quit input and empty the input buffer.

Use: Pressing the **DOWN ARROW** key empties the input buffer, but leaves the template unchanged. Pressing the **DOWN ARROW** key also prints a backslash (\) and turns insert mode off if it is on.

The cursor is positioned at the beginning of the line. Pressing the **F3** key copies the template to the input buffer just as the line was before the **DOWN ARROW** key was pressed.

For example assume the display shows:

```
1:*This is a sample file.  
2:*[]
```

Assume you want to replace the line. Type the phrase **Sample File**.

```
1:*This is a sample file.  
Sample File 2:*Sample File[]
```

Now, to reedit the line, press the **DOWN ARROW** key:

```
1:*This is a sample file.  
< DOWN ARROW> 2:*Sample File  
2:[]
```

The **RETURN** key can now be pressed to keep the original line or to perform any other intraline editing functions. If the **F3** key is pressed, the original template is copied to the input buffer:

```
< F3> 2:This is a sample file.[]
```

INS (Insert Mode)

Function: Enter insert mode

Use: Pressing the **INS** key causes entry into the insert mode. In the insert mode you can insert characters into your text. The current position in the template is not changed.

The cursor moves as each character is inserted. When you have finished inserting characters, however, the cursor is positioned at the same character as it was before the insertion began. Thus, characters are inserted before the character to which the cursor points.

For example assume the display shows:

```
1:*This is a sample file.  
2:*[]
```

Assume you press the **F2** key and type the letter **p**. Your display should look like this:

```
1:*This is a sample file  
< F2> 2:*This is a sam[]
```

Now press the **INS** key and insert the three characters **s**, **o**, and **n**.

```
1:*This is a sample file.  
< F2> p 2:*This is a sam[]  
< INS> son 2:*This is a samson[]
```

If you now press the **F3** key, the rest of the template is copied to the line:

```
1:*This is a samson[]  
< F3> 2:*This is a samsonple file.[]
```

If you were to press the **RETURN** key instead, the remainder of the template would be truncated, and the input buffer ended at the end of the insert:

< INS> son< RETURN> 1:★This is a samson

To exit the insert mode, use the **RIGHT ARROW**, **LEFT ARROW**, or **F3** key.

F5

Function: Create a new template.

Use: Pressing the **F5** key copies the current contents of the input buffer to the template. The contents of the old template are then destroyed. Pressing the **F5** key also prints an at sign (@), positions the cursor at the beginning of the next line, empties the input buffer, and turns the insert mode off if it is on.

NOTE

The **F5** key performs the same functions as the **DOWN ARROW** key, except that the template is changed and an at sign (@) is printed instead of a backslash (\).

For example assume the display shows:

```
1:★This is a sample file.  
2:★[]
```

Copy all characters up to the letter **p** by pressing the **F2** key followed by typing the letter **p**. Now press the **INS** key and type the letters **son**.

```
< F2> p      1: *This is a sample file.  
< INS> son    2: *This is a sam[]  
              2: *This is a samson[]
```

Now press the **INS** key again to return to the replace mode. Type the letters **ite** then press the **F3** key to copy the remainder of the line.

```
< INS> ite    2: *This is a samsonite[]  
< F3>         2: *This is a samsonite file.
```

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

```
< F3> 2: *This is a samsonite file.@  
      []
```

Additional editing can now be done using the above new template.

INTERLINE COMMANDS

Interline commands perform editing functions on entire lines. The following table summarizes each of the interline commands and their functions. More detailed command descriptions and examples of how each are used are included later in this chapter.

Command	Purpose
< line>	Edit Line
A	Append Lines
D	Delete Lines
E	End Editing
I	Insert Text
L	List Text
Q	Quit Editing
R	Replace Text
S	Search Text
W	Write Lines

Parameters

Each interline command, except Quit and End, accepts some optional parameter. The following is a list of these parameters with the form in which they will appear. The effect of an individual parameter depends on the command.

Parameter	Definition
line	Line indicates a line number to be entered by you. Line numbers must be separated by a comma or a space from other line numbers, other parameters, and other commands.

Line may be specified by any of the following ways:

A Number any integer less than 65 529.
If a number larger than the largest existing line number is specified, then line indicates the line after the last line number.

A Period (.) If a period is specified for line, then line indicates the current line number. The current line is the last line edited, and not necessarily the last line displayed. The current line is marked on your display by an asterisk (*) between the line number and the first character.

Number Sign (#) The number sign indicates the line after the last line number. Specifying # for line has the same effect as specifying a number larger than the last line number.

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

? The question mark parameter directs EDLIN to ask the user if the correct string has been found. The question mark is used only with the Replace and Search commands. Before continuing, EDLIN waits for you to type a **Y** or to press **RETURN** key 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 parameter is used only with the Search and Replace commands.

Each string must be terminated by pressing and holding the **CTRL** key and then typing a **Z** or pressing the **RETURN** key (see the Replace command for details). No spaces should be left between a string and its command letter, unless you want spaces as part of a string.

Edit

Syntax: [line]

Function: Edit line

Use: When a line number is entered, EDLIN displays the line number and text, then, on the line below, reprints the line number. The line is then ready for editing. You may use any of the available intraline 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 entered (that is, 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, you can press the **RETURN** key to accept the line as it is as long as the cursor is at the beginning or the end of the line.

NOTE

If the **RETURN** key is pressed while the cursor is in the middle of the line, the remainder of the line is truncated.

Assume the following file exists and is ready to edit:

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

To edit line 4, enter:

```
4
```

The contents of the line are displayed along with a cursor below the line:

```
4:*four.  
4:*[]
```

Now press the **INS** key and type the word **number**. Press the **F3** key to copy the remainder of the line.

```
< INS> number      4:number[]  
< F3>< RETURN> 4:number four.
```


Append

Syntax: [n]A

Function: Append lines from the input file to the editing buffer.

Use: Use this command for extremely large files that will not fit into memory all at one time. By writing out part of the editing buffer to the output file with the Write command (discussed later in this chapter), room is made for lines to be appended with the Append command.

If A is typed without a parameter, lines are appended to the part of the file currently in memory until available memory is three-fourths full or until there are no more lines to append.

Use the Write command to write lines to the output file. If the parameter n is given, then n lines are appended to that part of the file that currently is in memory. If n is not given, then as much of the input file as possible is read into the editing buffer until the editing buffer is three-fourths full.

Delete

Syntax: [line][,line] D

Function: Delete the specified lines and all lines in between.

Use: If the first line is omitted, the first line defaults to the current line (the line with the asterisk next to the line number). If the second line is omitted, then just the first line is deleted.

When lines have been deleted, the line immediately after the deleted section becomes the current line. The current line then has the same line number as the first line specified in the command parameter had before the deletion occurred.

For example assume the following file exists and is ready to edit:

```
1: This is a sample file.  
2: Use: to demonstrate dynamic line numbers  
3: See what happens when you  
4: Delete and Insert  
.  
.  
.  
25: (The D and I commands)  
26: (Use <SHIFT-BRK> to exit insert mode)  
27:* Line numbers
```

To delete multiple lines, enter line, line D as in the example shown.

5,24 D

The result is:

```
1:  This is a sample file.
2:  Use: to demonstrate dynamic line numbers
3:  See what happens when you
4:  Delete and Insert
5: * (The D and I commands)
6:  Use <SHIFT-BRK> to exit insert mode
7:  Line numbers
```

To delete a single line, enter:

6D

The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
6:* Line numbers
```

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

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3:* See what happens when you
4: Delete and Insert
5: (The D and I commands)
6: (Use <SHIFT-BRK> to exit insert mode)
7: Line numbers
```

To delete beginning with the current line, enter:

,6D

The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3:* Line numbers
```


End

Syntax: E

Function: End the editing session

Use: This command causes the edited file to be saved on diskette, renames the original input file "filename.BAK," and then exits EDLIN to the MS-DOS command level. If the file is created during the editing session, no .BAK file is created.

The END command takes no parameters. Therefore, you cannot tell EDLIN on which drive to save the file. The drive must be selected when the editing session is invoked. If the drive is not designated when EDLIN is invoked, the file is saved on the diskette in the default drive.

If you don't specify a drive when EDLIN is invoked, and want to save the file on a drive other than the default drive, it is still possible to use the COPY command to copy the file to a different drive.

You must be sure that the diskette contains enough free space for the entire file to be written. If the diskette does not contain enough free space, the write is aborted and the edited file lost, although part of the file may be written out.

The only possible command is:

E< RETURN>

After execution of the END command, control is returned to MS-DOS.

Insert

Syntax: [line] I

Function: Insert line(s) of text immediately before the specified line.

Use: If you are creating a new file, the I command must be given before text can be inserted. In this case, the insert begins with line number 1.

EDLIN remains in insert mode until you press and hold the **CTRL** key and then type a **Z** or until you press and hold the **SHIFT** key and then press the **BRK/PAUS** key. Successive line numbers appear automatically each time the **RETURN** key is pressed.

When you have finished inserting and have exited the insert mode, the line specified in the command which now immediately follows the inserted lines, becomes the current line. All line numbers following the inserted section are incremented by the number of lines inserted.

If the line is not designated in the command, the default is the current line number (the lines are inserted immediately before the current line).

If the line is an integer larger than the last line number of your file, or if you specify the number sign (#) as the line, inserted lines are appended to the end of the file. In this case, the last line inserted becomes the current line. (This is the same as when the file is being created.)

For example assume the following file exists and is ready to edit:

```
1: This is a sample file.  
2: Use: to demonstrate dynamic line numbers  
3: See what happens when you  
4: Delete and Insert  
5: (The D and I commands)  
6: (Use <SHIFT-BRK> to exit insert mode)  
7:★ Line numbers
```

To insert text before a specific line (not the current line), type the line number and **I**.

```
4 I
```

The result is:

```
4: [ ]
```

Now, enter the new text for lines 4 and 5:

```
4: experiment with  
5: those very useful commands that
```

Then to end the insertion, press and hold the **CTRL** key, type **Z**, and then press the **RETURN** key.

Now type **L** to list the file; the result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: experiment with
5: those very useful commands that
6:* Delete and Insert
7: (The D and I commands)
8: (Use <SHIFT-BRK> to exit insert mode)
9: Line numbers
```

To insert lines immediately before the current line, enter:

I

The result is:

6:[]

Now, insert the following text, press and hold the **CTRL** key, and type a **Z**.

6: perform the two major editing functions,

Now to List the file and see the result, type:

L

The result is:

```
1: This is a sample file.  
2: Use: to demonstrate dynamic line numbers  
3: See what happens when you  
4: experiment with  
5: those very useful commands that  
6: perform the two major editing functions,  
7: * Delete and Insert  
8: (The D and I commands)  
9: (Use <SHIFT-BRK> to exit insert mode)  
10: Line numbers
```

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

```
11 I
```

This produces the following:

```
11:[]
```

Now, enter the following new lines:

```
11: The insert command can place new lines  
12: anywhere in the file; there's no space problems,  
13: because the line numbers are dynamic;  
14: They'll slide all the way to 65 529.
```

End insertion by pressing and holding the **SHIFT** key and pressing the **BRK/PAUS** key. The new lines will appear at the end of all previous lines in the file. Now enter the list command:

L

The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: experiment with
5: those very useful commands that
6: perform the two major editing functions,
7: Delete and Insert
8: (The D and I commands)
9: (Use <SHIFT-BRK> to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space problems.
13: because the line numbers are dynamic;
14: They'll slide all the way to 65 529.
```


List

Syntax: < line> < ,line> L

Function: List the specified range of lines, including the two lines designated in the command.

Use: The List command operates four different ways according to the following guidelines.

* L

If you want to view the lines immediately before and after the line you are currently working on, type in the command L. L lists the 11 lines before the current line, the current line, and the 11 lines following the current line. For example, if the current line is 15, lines 4 to 26 are displayed. Note that the current line is denoted by an asterisk.

```
4: Delete and Insert
5: (The D and I commands)
.
.
.
15:* The current line contains an asterisk.
.
.
.
26: (Use <SHIFT-BRK> to exit insert mode)
```

*** < line> , < line> L**

If you want to view a block of lines that doesn't include the current line, specify the beginning and ending lines of that block. For example, the command 2, 5 L displays lines 2 to 5. If you specify a block longer than 23 lines, all the lines are displayed but only the last 24 stay on the display. To view the first lines before they run off the screen, you must use the **BRK/PAUS** key.

- 2: Use: to demonstrate dynamic line numbers
- 3: See what happens when you
- 4: Delete and Insert
- 5: (The D and I commands)

*** < line> , L**

If you want to view a line and the block of lines following it, specifying only the starting line displays the starting line and 22 lines following it. For example, the command 13, L displays lines 13 to 35.

- 13: The specified line is listed first in the range.
- 14: The current line remains unchanged by the L command.
- 15:* The current line contains an asterisk.
- .
- .
- .
- 35: <SHIFT-BRK> exits interline insert command mode.

***, < line> L**

If you want to view a line and the block of lines immediately preceding it, specifying only the ending line displays the block of lines starting 11 lines before the current line and ending with the line you specify. For example, if the current line is 20, the command, 27 L displays lines 9 to 27. This rule has two exceptions. If the block of lines specified is longer than 24 lines, all but the last 24 lines scroll off the top of the display. Also, if the ending line you specify is before the assumed starting line, the command is executed as though you only entered the command L (see above).

2: Use: to demonstrate dynamic line numbers

3: See what happens when you

4: Delete and Insert

5: (The D and I commands)

.

.

.

20:* The current line contains an asterisk.

.

.

.

26: (Use <SHIFT-BRK> to exit insert mode)

27: Line numbers.

QUIT

Quit

Syntax: Q

Function: Quit (abort) the editing session, do not save any editing changes, and exit to the MS-DOS operating system.

Use: No .BAK file is created. The Quit command takes no parameters. It is simply a fast means of aborting an editing session. As soon as the Quit command is given, EDLIN displays the message:

Abort edit (Y/N)?[]

Type **Y** to abort the editing session. If you decide to continue the editing session, type **N**.

For example assume the following file exists and is ready to edit:

```
1: This is a sample file.
2: Use: to demonstrate dynamic file numbers
3: Compare the before and after
4: See what happens when you
5: Delete and Insert
6: Line numbers
```

Now, to delete line 3, enter:

3 D

To list the file, enter:

L

```
1: This is a sample file.  
2: Use: to demonstrate dynamic line numbers  
3: See what happens when you  
4: Delete and Insert  
5: Line numbers
```

Now, to delete the changes and to quit the editing session, enter:

Q

The result is:

```
Abort edit (Y/N)?[]
```

Enter **Y** to exit to the operating system command level:

```
Abort edit (Y/N)?Y  
A:[]
```

Replace

- Syntax:** [line][,line][?]R < string1> < CONTROL-Z>
< string2>
- Function:** Replace all occurrences of string1 in the specified range with string2.
- Use:** As each occurrence of string1 is found, it is replaced by string2. Each line in which a replacement occurs is displayed. If a line contains two or more replacements of string1 with string2, then the line is displayed once for each occurrence. When all occurrences of string1 in the specified range are replaced by string2, the Replace command terminates and the asterisk prompt (*) reappears.

The following information applies to the specification of string1, the string that is to be replaced, and string2, the replacement string.

- If a second string is to be given as a replacement, the string1 must be terminated by pressing and holding the **CTRL** key and then typing **Z**. If the string2 is to be omitted, then string1 may be terminated by pressing and holding the **CTRL** key and then typing a **Z** followed by pressing the **RETURN** key. String2 must be terminated the same way.
- If string1 is omitted, the replacement is terminated. If string2 is omitted, then string1 is deleted from all lines in the range.

The range is designated by the first two parameters in the filespec. Use line numbers to specify a replacement range. The following information applies when you are specifying a replacement range.

- If the first `< line>` is omitted in the range argument (as in `,< line>`), then the first `< line>` defaults to the line after the current line.
- If the second `< line>` is omitted (as in `< line>` or `< line>,`), the second `< line>` defaults to `#` where `#` indicates the line after the last line of the file. This is equivalent to `< line>,#`.

The question mark (?) parameter is optional. If it is present, the Replace command stops at each line with a string that matches string1, the line is displayed with string2 in place of string 1, and the following prompt is displayed.

O.K.?

If you type a **Y** or press the **RETURN** key, then string2 replaces string1 and the next occurrence of string1 is found. Again the “O.K.?” prompt is displayed. This process continues until the end of the range or until the end of the file. After the last occurrence of string1 is found, EDLIN returns the asterisk prompt (*).

If you press any key besides **Y** or the **RETURN** key after the “O.K.?” prompt, the string1 is left as it was in the line, and the Replace command continues to the next occurrence of string1.

If string1 occurs more than once in a line, each occurrence of string1 is replaced individually, and the "O.K.?" prompt is displayed after each replacement. In this way, only the desired string1 is replaced, and you prevent replacement of embedded strings.

For example assume the following file exists and is ready for editing.

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: experiment with
5: those very useful commands that
6: perform the two major editing functions,
7: Delete and insert
8: (The D and I commands)
9: (Use <SHIFT-BRK> to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space problem.
13: because the line numbers are dynamic.
14: They'll slide all the way to 65 533.
```

To replace all occurrences of string1 with string2 in a specified range, enter:

2,12 Rand<CTRL-Z> or <RETURN>

The result is:

```
5:  those very useful commors that
7:  Delete or Insert
8:  (The D or I commands)
8:  (The D or I commors)
11: The insert commor can place new lines
```

Note that in the above replacement, some unwanted substitutions have occurred. To avoid these, and confirm each replacement, the same original file can be used:

```
.
.
.
5:  those very useful commands that
.
.
.
7:  Delete and Insert
8:  (The D and I commands)
.
.
.
11: The insert command can place new lines
.
.
.
```

only with a slightly different command. This time, to replace only certain occurrences of the first string with the second string, enter:

2? Rand< CTRL-Z> or < RETURN>

The result is:

```
5:  those very useful commors that
0.K.? N
7:  Delete or Insert
0.K.? Y
8:  (The D or I commands)
0.K.? Y
8:  (The D or I commors)
0.K.? N
11: The insert commor can place new lines
0.K.? N
*[]
```

Now, enter the List command (L) to see the result of all these changes:

```
.  
. .  
. .  
5:  those very useful commands that  
. .  
. .  
7:  Delete or Insert  
8:  (The D or I commands)  
. .  
. .  
11: The insert command can place new lines  
. .  
. .
```

Search

Syntax: [line][,line] [?] S< string>

Function: Search the specified range of lines for the specified string.

Use: The string must be terminated by pressing the **RETURN** key. The first line that matches string is displayed and becomes the current line. The Search command terminates when a match is found. If no line contains a match for string, the message "Not found" is displayed.

If the optional parameter, question mark (?), is included in the command, the first line with a matching string is displayed. The following prompt is then displayed.

O.K.?

If you type the letter **Y** or press the **RETURN** key, the line becomes the current line and the search terminates. If you press any other key, the search continues until another match is found, or until all lines have been searched. The “Not found” message is then displayed.

The following guidelines apply to the specification of Search command parameters.

- If the first line is omitted (as in ,< line> S< string>), the first line defaults to the line after the current line.
- If the second line is omitted (as in < line> S< string > or < line> , S< string>), the second line defaults to #, which is the same as < line> ,# S< string> .
- If string is omitted, no search is made and the command terminates immediately.

For example assume the following file exists and is ready for editing.

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: experiment with
5: those very useful commands that
6: perform the two major editing functions,
7: Delete and Insert
8: (The D and I commands)
9: (Use <SHIFT-BRK> to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space problems.
13: because the line numbers are dynamic;
14: *They'll slide all the way to 65 529.
```

To search for the first occurrence of a string, enter:

```
2,12 Sand< RETURN>
```

The result is:

```
5: those very useful commands that
```

To get the “and” in line 7, modify the search command by entering:

```
< DEL>< F3>, 12 Sand< RETURN>
```

The search then continues from the line after the current line (line 5), since no first line is given. The result is:

```
7: Delete and Insert
```

WRITE

To search through several occurrences of a string until the correct string is found, enter:

1, ? Sand

The result is:

5: those very useful commands that
O.K.?_

If you press any key except **Y** or the **RETURN** key the search continues; so type **N** here:

O.K.? N

Continue:

7 Delete and Insert
O.K.?[]

Now press **Y** to terminate the search.

O.K.? Y
*[]

Write

Syntax: [n]W

Function: Write lines from the editing buffer to the output file.

Use: The Write command is used when you are editing files that are larger than available memory. Whenever you edit a file it is loaded into a temporary storage place in memory called the editing buffer. By executing the Write command, lines are written from the input buffer to your diskette. This leaves room in the input buffer for more lines.

If W is typed with no n parameter, then lines are written until memory is one-fourth full. If the n parameter is given, then n lines are written to your diskette.

Note that lines are written to diskette beginning with the start of the file; subsequent lines in the editing buffer are renumbered beginning with one. A later Append command will append lines to any remaining lines in the editing buffer.

ERRORS WHEN INVOKING EDLIN

EDLIN error messages occur either when you try to invoke EDLIN or during the actual editing session.

Cannot edit .BAK file—rename file

Cause: An attempt was made to edit a file with the filename extension .BAK. .BAK files cannot be edited because the 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 using 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 the user specified an illegal diskette drive or an illegal filename.

Cure: Check the EDLIN invocation command line for an illegal filename and illegal diskette drive entries. If the command is no longer on the display and if the user has not yet entered a new command, the EDLIN invocation command can be recovered by pressing the **F3** key.

If the invocation command line contains no illegal entries, run the CHKDSK program for the specified diskette drive. If the status report shows the diskette directory full, remove the diskette and insert and format a new diskette. If the CHKDSK status report shows the diskette directory is not full, check the EDLIN invocation command for an illegal filename or illegal diskette drive designation.

ERRORS WHILE EDITING

Entry Error

Cause: The last command entered contained a syntax error.

Cure: Reenter the command with the correct syntax.

Line too long

Cause: During Replace command mode, the string given as the replacement string causes the line to expand beyond the limit of 254 characters. EDLIN aborts the Replace command.

Cure: Divide the long line into two lines, then retry the Replace command.

Disk Full—file write not completed

Cause: The End command was given, but the diskette did not contain enough free space for the whole file. EDLIN aborts the End command and returns the user to the operating system. Some of the file may have been written to the diskette.

Cure: Only a portion (at most) of the file is saved. You should probably delete whatever file was saved and restart the editing session. None of the file not written is available after this error. Always be sure that the diskette has sufficient free space for the file to be written before you begin your editing session.

5

Debugging (DEBUG)

Invocation	5-3
Commands	5-5
Parameters	5-5
COMPARE	5-10
DUMP	5-10
ENTER	5-12
FILL	5-14
GO	5-15
HEX	5-16
INPUT	5-17
LOAD	5-17
MOVE	5-19
NAME	5-20
OUTPUT	5-23
QUIT	5-24
REGISTER	5-24
SEARCH	5-27
TRACE	5-28
UNASSEMBLE	5-29
WRITE	5-31
Command Summary	5-33
Error Messages	5-34

DEBUG is a debugging program that provides a controlled testing environment for binary and executable object 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 computer register, and then to immediately reexecute a program to check the validity of the changes.

All DEBUG commands may be aborted at any time by pressing and holding the **SHIFT** key and pressing the **BRK/PAUS** key. The **BRK/PAUS** key suspends the display, so that you can read it before the output scrolls away. Pressing any alphabetic key or the **RETURN** key (including the **BRK/PAUS** key again) restarts the display. All of these commands are consistent with the control-key functions available at the MS-DOS command level.

INVOCATION

To invoke DEBUG, enter:

```
DEBUG [< filespec> [< arglist> ]]
```

For example, if a filespec is specified, then the following is a typical invocation:

DEBUG FILE.EXE

DEBUG 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. The DEBUG prompt is a right angle bracket (>).

An arglist may be specified if filespec is present. These are 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 invoked 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.

If no filespec is specified, then DEBUG is invoked as follows:

DEBUG

DEBUG then returns with the prompt, signaling that it is ready to accept your commands. The registers and flags are set to the following values for the program being debugged:

- The segment registers (CS, DS, ES, and SS) are set to the bottom of free memory; that is the first segment after the end of the DEBUG program.
- The instruction pointer (IP) is set to 100 hexadecimal
- The stackpointer (SP) is set to the end of the segment, or the bottom of the transient portion of COMMAND.COM, whichever is lower. The segment size at offset 6 in the program segment header is reduced by 100 hexadecimal to allow for a stack of that size.
- The remaining registers (AX, BX, CX, DX, BP, DI, and SI) are set to zero. However, if you invoke DEBUG with a filespec (or Name a file and Load it), the BX and CX registers contain the length of the file in bytes. BX has the high portion.
- The flags are set to their cleared values (see the Register command).
- The default disk transfer segment is set to 80 hexadecimal in the code segment.
- If an .EXE file was loaded, DEBUG performs any necessary relocation and sets the segment registers, stackpointer, and instruction pointer to the values defined in the file. However the DS and ES registers point to the program segment header at the lowest available segment. The BX and CX registers contain the size of the program (smaller than the file size because of the relocation header).

COMMANDS

Each DEBUG command consists of a single letter followed by one or more parameters. The control keys and the special editing functions described in Chapter 2 apply inside DEBUG as well.

If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with a caret (^) and the word “Error.”

For example:

```
>dcs:100 cs:110
      ^Error
>[]
```

All commands and parameters may be entered in either uppercase or lowercase letters. Any combination of uppercase and lowercase letters may be used in commands.

PARAMETERS

All DEBUG commands accept parameters except the Quit command. All numerical parameters (addresses and so on) are entered in hexadecimal form, therefore, the legal characters for these parameters are 0–9 and A–F. 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
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 into or read from an address or register.
sector	A one- to three-digit hexadecimal value used to indicate the logical sector number on the diskette and the number of diskette sectors to be written or loaded. Diskettes are formatted with 8 physical sectors (numbered 1–8) per track (numbered from 0) per side (0 or 1).

For a single-sided diskette, the logical sectors are sequentially numbered: Track 0, Sectors 1–8; Track 1, Sectors 1–8; and so on.

For double-sided diskettes, the sectors on the second side are counted before moving to the next track: Track 0, Side 0, Sectors 1–8; Track 0, Side 1, Sectors 1–8; Track 1, Side 0, Sectors 1–8; and so on.

In DEBUG (and when using INT 25 and 26 for direct-disk access), the logical sector numbering depends on the last file or directory access by MS-DOS. If the diskette was single sided when that access took place, then for subsequent accesses by DEBUG, the logical sectors are numbered from 0 to 13F hexadecimal, starting at Track 0, Sector 1, Side 0; logical sector 8 would be Track 1, Sector 1, Side 0, and so on.

If it was double sided, the logical sectors are numbered from 0 to 27F hexadecimal and are calculated in the same way as MS-DOS accesses a double-sided diskette: logical sector 0 is Track 0, Sector 1, Side 0; logical sector 7 is Track 0, Sector 8, Side 0; logical sector 8 is Track 0, Sector 1, Side 1; and so on.

In other words, each track has 16 absolute sectors, 8 on each side. A simple way to ensure that you access a diskette correctly is to do a directory command on a diskette of the desired type (single or double sided) before invoking DEBUG.

The maximum number of sectors that may be loaded or written with a single command is 80 hexadecimal. Also, a command is not allowed to transfer sectors across a 64K segment boundary. This means, for example, that you could load 40 hexadecimal sectors at 0:8000, but not at 0:8001. Similarly, you could load 80 hexadecimal sectors at 800:0000 but not at 800:0001. Any attempt to violate these limitations will result in a "Disk error" error message.

value A hexadecimal value up to four digits used to specify a port number or the number of times a command function is to be repeated.

address A two-part designation consisting of either an alphabetic segment register designation or a four-digit segment address and 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  
053A:0100
```

The colon is required between a segment designation and an offset.

range

Can be expressed in either of the following forms:

- As two addresses: < address> < address>
- As one address, an L, and a value: < address>
L < value> where value is the number of bytes the command should operate on.

Examples:

```
CS:100 110  
CS:100 L 10
```

The following is illegal:

```
CS:100 CS:110  
          ^ Error
```

The limit for range is 10000 hexadecimal. To specify a value of 10000 hexadecimal within four digits, enter 0000 (or 0).

list

A series of byte values or of strings. List must be the last parameter on the command line.

Example:

```
fcs:100 42 45 52 54 41
```

string

Any number of characters enclosed within quotation marks. Quotation marks may be either single (') or double ("). Within strings, the opposite set of quotation marks may be used freely as literals. If the delimiter quotation marks must appear within a string, the quotation 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.'

These following strings are legal.

""This is a "string" is okay.""
""This is a ""string"" is okay.""

However, the following string is illegal.

""This is a "string" is not.""

Note that the double quotation 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.

COMPARE

COMPARE

Syntax: C<range> <address>

Function: Compare the portion of memory specified by range to a portion of the same size beginning at address.

Use: If the two areas of memory are identical, there is no display and DEBUG returns with its prompt. If there are differences, they are displayed as:

<address1> <byte1> <byte2> <address>

Example: The following commands have the same effect.

C100,200 300

C100L100 300

In each case the block of memory from 100 to 200 hexadecimal is compared with the block of memory from 300 to 400 hexadecimal.

DUMP

Syntax: D[address]
D[range]

Function: Display the memory contents of either a single address, a range of addresses, or the number of lines specified by value beginning at the address specified.

Use: The Dump command operates according to the following guidelines:

- If a single address only is specified, the contents of 128 bytes are displayed.

-
- If a range of addresses is specified, the contents of the range are displayed.
 - If the Dump command is entered without parameters, the result is the same as if you had specified a single address, except that the display begins at the current location in the DS (data segment).

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 begins on a 16-byte boundary, and shows 16 bytes with a hyphen between the eighth and ninth bytes. The first line may have fewer than 16 bytes if the starting address is not on a boundary; in this case, the second line will begin on a boundary.

Example: Assume you enter the following command:.

dcS:100 110

DEBUG displays:

053A:0100 42 45 52 54 41 ... 4E 44 BERTA T. BORLAND

If the following command is entered:

D

the next 128 bytes will be displayed in the format described above. Each line of the display begins with an address incremented by 16 from the address on the previous line. Each subsequent D (entered without parameters) displays the bytes immediately following those last displayed.

If you enter the command:

DCS:100 L 20

the display is formatted as described above, but only 20 hexadecimal bytes (2 lines) will be displayed.

If you enter the command:

DCS:100 115

the display is formatted as described above, but all the bytes in the range of lines from 100 to 115 hexadecimal in the CS segment are displayed.

ENTER

Syntax: E< address> [list]

Function: Display and change the contents of individual memory locations.

Use: If the optional list of hexadecimal values is entered, the replacement of byte values occurs automatically. If there was an error on the command line, no byte values are changed.

If the address is entered without the optional list, DEBUG displays the address and its contents, and waits for you to perform one or more of the following actions:

- Replace a byte value with a value you type in after the current value. If the value typed is not a legal hexadecimal value or if more than two digits are typed, the illegal or extra character is not echoed.

-
- Press the space bar to advance to the next byte. To change the value, simply enter the new value as described above. If you space beyond an eight-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.
 - Type a hyphen (-) to return to the preceding byte. If you decide to change a byte after the current position, type a hyphen to return the current position to the previous byte. When the hyphen is typed, a new line is started with the address and its byte value is displayed.
 - Press the **RETURN** key to terminate the Enter command. The **RETURN** key may be pressed at any byte position.

Example: Assume the following command is entered.

ECS:100

DEBUG displays:

053A:0100 EB.[]

To change this value to 41, type **41** as shown.

053A:0100 EB.41[]

To step through the subsequent bytes, press the space bar to see:

053A:0100 EB.41 10. 00. BC.[]

To change BC to 42:

053A: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), and then replace 10 with 6F:

```
053A:0100 EB.41 10.00. BC.42-  
053A:0102 00.-[]  
053A:0101 10.6F[]
```

Pressing the **RETURN** key ends the Enter command and returns you to the DEBUG command level.

FILL

Syntax: F<range> <list>

Function: Fill the addresses in the range with the values in the list.

Use: The Fill command operates according to the following guidelines:

- If the range contains more bytes than the number of values in the list, the list is 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 are ignored.
- If you enter only an offset for the starting address of the range, DS is used as the segment.

Example: Assume the following command is entered.

```
F053A:100 L 100 42 45 52 54 41
```

DEBUG fills memory locations 053A:100 through 053A:200 with the bytes specified. The five values are repeated until all 100 hexadecimal bytes are filled.

GO

Syntax: G[=< address> [< address> ...]]

Function: Execute the program currently in memory.

Use: The Go command operates according to the following guidelines:

- If the Go command is entered alone, the program begins execution at the current CS:IP (CS register: instruction pointer). If the Go command immediately follows DEBUG invocation, the program will execute as if it had not been run in DEBUG mode.
- If =address is set, execution begins at the address specified. If the segment designation is omitted, only the instruction pointer is set (the current value of CS is used).
- If the segment designation is included in =address, both the CS register and the instruction pointer are set.
- When an address is entered without the equal sign, it causes a breakpoint to be set (you may set up to ten). When the instruction at the specified address is reached (breakpoint), DEBUG displays the registers, flags, and the next instruction to be executed. If you now enter another Go command, execution begins at the instruction after the breakpoint.

Breakpoints may be set only at addresses containing the first byte of an 8088 opcode. This byte is replaced with an interrupt opcode (CC hexadecimal). The current flags, CS, and IP are pushed onto your stack, and an IRET instruction is used to jump to your program at the specified entry point. This requires a valid stackpointer with six bytes available for the Go command to operate correctly.

When the breakpoint interrupt opcode (CC hexadecimal) is executed, all breakpoint addresses are restored to their original instruction opcodes. If execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions. Note that since the breakpoint mechanism depends on the temporary replacement of the original instruction, you cannot set breakpoints in a program stored in read-only memory (ROM).

Example: Assume the following command is entered:

GCS:7550

The program currently in memory executes up to the address 7550 in the CS segment. Then DEBUG displays registers and flags, after which the Go command is terminated.

HEX

Syntax: H< address> < address>

Function: Perform hexadecimal arithmetic on the two parameters.

Use: DEBUG adds the two parameters, and 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 the following command is entered.

H10A 19F

DEBUG performs the calculations and then returns the results:

02A9 0095

INPUT

Syntax: I< value>

Function: Input and display one byte from the port specified by value.

Use: A 16-bit port address is allowed.

Example: Assume the following command is entered.

I2F8

Assume also that the byte at the port is 42 hexadecimal. DEBUG inputs the byte and displays the value:

42

LOAD

Syntax: L[< address> [< drive> < sector> < sector>]]

Function: Load a file or diskette sectors into memory.

Use: If this command is to be used to load a file, the file must have been named either with the DEBUG invocation command or with the Name command. Both the invocation and the Name commands format a filename into the FCB at CS:5C.

The Load command operates according to the following guidelines:

- If the Load command is given without any parameters, the Named file is loaded into memory beginning at address CS:100. BX:CX is set to the number of bytes loaded.

-
- If the Load command is given with an address parameter, loading begins at the address specified. If only an offset is entered, the CS register is used for the segment.
 - If the Load command is entered with all parameters, absolute diskette sectors are loaded, not a file. The sectors are taken from the drive specified (the drive designation is numeric here—0=A, 1=B, 2=C, 3=D, and so on). DEBUG begins loading with the first sector specified and continues until the number of sectors specified in the second sector have been loaded. See the parameter description section for an explanation of sector numbering and memory limitations.

Example: Assume the following commands are entered.

```
A:DEBUG  
> NFILE.COM
```

Now, to load FILE.COM, enter:

```
L
```

DEBUG loads the file and returns the DEBUG prompt. Assume you want to load only portions of a file or certain sectors from a diskette. To do this, enter:

```
L053A:100 2 0F 6D
```

DEBUG then loads 109 (6D hexadecimal) sectors beginning with logical sector number 15 (0F hexadecimal) into memory beginning at address 053A:0100. When the sectors have been loaded, DEBUG returns its prompt. If any disk errors are reported, check the syntax and retry the command.

If the file has a .EXE extension, then it is relocated to the load address specified in the header of the .EXE file—the address parameter is always ignored for .EXE files. Note that the header itself is stripped from the .EXE file before it is loaded into memory. Thus the size of an .EXE file on diskette differs from its size in memory (BX:CX reflects the size in memory). If you need to load the .EXE file into memory without relocation, simply rename it with a different extension (such as .XXX) before loading it.

If the file named by the Name command or specified on invocation is a .HEX file, then entering the Load command with no parameters causes loading of the file beginning at the address specified in the .HEX file.

If the Load command includes the option address, DEBUG adds the address specified in the Load command to the address found in the .HEX file to determine the start address for loading the file.

MOVE

Syntax: M< range> < address>

Function: Move the block of memory specified by range to the location beginning at the address specified.

Use: Overlapping moves (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. For moves from higher addresses to lower addresses, the move sequence begins with the data at the block's lowest address and works towards the highest.

NAME

For moves from lower addresses to higher addresses, the move sequence begins with the data at the block's highest address and works 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; that is, the Move command really 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 you enter:

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 enter the Dump command, using the address entered for the Move command, to review the results of the move.

NAME

Syntax: N< filename> [< filename> ...]

Function: Set filenames.

Use: The Name command performs two distinct functions, both having to do with filenames. First, the Name command is used to assign a filename for a later Load or Write command. Thus, if you invoke DEBUG without naming any file to be debugged, then the Name command must be given 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 functions overlap. Consider the following set of DEBUG commands:

```
>NFILE1.EXE  
>L  
>G
```

Because of the two-pronged effect of the Name command, the following happens:

1. The Name command assigns the filename FILE1.EXE to the filename to be used in any later Load or Write commands.
2. The Name command also assigns the filename FILE1.EXE to the first filename parameter to be used by any program that is later debugged.
3. The Load command loads FILE1.EXE into memory.
4. The Go command causes FILE1.EXE to be executed with FILE1.EXE as the single filename parameter that is, FILE1.EXE is executed as if FILE1 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, the Name command 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 command.

Four distinct regions of memory are 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 entered

An FCB for the first filename parameter given to the Name command is set up at CS:5C. If a second filename parameter is given, 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 would be:

```
DEBUG PROG.COM  
-NPARAM1 PARM2/C  
-G  
-
```

In this case, the Go command executes the file in memory as if the following command line had been entered.

```
PROG PARAM1 PARAM2/C
```

Testing and debugging, therefore, reflect a normal run-time environment for PROG.COM.

OUTPUT

Syntax: O< value> < byte>

Function: Send the byte specified to the output port specified by value.

Use: A 16-bit port address is allowed.

Example: Enter:

```
O2F8 4F
```

DEBUG outputs the byte value 4F to port 2F8.

QUIT

QUIT

Syntax: Q

Function: Terminate the debugger.

Use: The Quit 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, enter:

Q< RETURN>

DEBUG is terminated, and control returns to the MS-DOS command level.

REGISTER

Syntax: R[register-name]

Function: Display the contents of one or more CPU registers.

Use: If no register-name is entered, the Register command displays the contents of all registers and flags.

If a register-name is entered, the 16-bit value of that register is displayed in hexadecimal, and then a colon appears as a prompt. You may then either enter a value to change the register, or simply press the **RETURN** key if no change is wanted.

The only valid register-names are:

AX	BP	SS	
BX	SI	CS	
CX	DI	IP	(IP and PC both refer to the instruction pointer.)
DX	DS	PC	
SP	ES	F	

Any other entry for register-name results in a BR error message.

If F is entered as the register-name, DEBUG displays the flags as a series of two-character alphabetic codes and a hyphen (-) prompt. To alter a flag (flags are either set or clear), enter the opposite two-letter code. You can alter multiple flags with one command, but not the same flag twice. They may be entered in any order and spaces are not required between flag entries. The alphabetic codes are listed in the following table:

Flag	Set	Clear
Overflow	OV	NV
Direction	DN Decrement	UP Increment
Interrupt	EI Enabled	DI Disabled
Sign	NG Negative	PL Plus
Zero	ZR	NZ
Auxiliary	AC	NA
Carry		
Parity	PE Even	PO Odd
Carry	CY	NC

To exit the Register 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 start-up, the segment registers are set to the bottom of free memory, the instruction pointer is set to 0100 hexadecimal, the stack pointer is set to 005A hexadecimal, all flags are cleared, and the remaining registers are set to zero.

Example:

Enter:

R

DEBUG displays all registers, flags, and the decoded instruction for the current location. If the location is CS:11A, then DEBUG might display:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000
SI=005C DI=0000 DS=053A ES=053A SS=053A CS=053A
IP=011A  NV  UP  DI  NG  NZ  AC  PE  NC
053A:011A  CD21                INT      21
```

If you enter:

RF

DEBUG displays the flags:

```
NV UP DI NG NZ AC PE NC - [ ]
```

Now enter 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, enter either the R or RF command:

```
RF
NV UP EI PL NZ AC PE CY - [ ]
```

Enter different flag values or just press the **RETURN** key to leave the flags this way.

SEARCH

Syntax: S< range> < list>

Function: Search the range specified for the list of bytes specified.

Use: 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 enter:

```
SCS:100 110 41
```

DEBUG might return the response:

```
053A:0104
053A:010D
>[ ]
```

TRACE

Syntax: T[=< address>][< value>]

Function: Execute one instruction and display the contents of all registers, flags, and the decoded instruction.

Use: If the optional =address is entered, tracing occurs at the address specified. If only an offset is entered, CS is used for the segment. The optional value causes DEBUG to execute and trace the number of steps specified by value. After each instruction is executed, the contents of all registers, flags, and the decoded instruction is displayed (this is the same sequence that occurs when a breakpoint is reached from the Go command). The Trace command uses the hardware trace mode of the 8088 microprocessor, so it is possible to trace instructions stored in ROM.

Example: Enter:

T

DEBUG returns a display of the registers, flags, and decoded instruction for that one instruction. Assume that the current position is 053A:011A; then DEBUG might return the display:

```
AX=0E00 BX=00FF CX=0007 DX = 01FF SP=039D BP=0000
SI=005C DI=0000 DS=053A ES=053A SS=053A CS=053A
IP=011A NV UP DI NB NZ AC PE NC
053A:011A CD21          INT      21
```

Now enter:

T=011A 10

DEBUG executes 10 hexadecimal 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 pressing the **BRK/PAUS** key suspends the display at any point, so that you can study the registers and flags for any instruction.

UNASSEMBLE

Syntax: U[address]
 U[range]

Function: Disassembles instructions and displays the assembly language mnemonics that correspond to them, along with addresses and byte values. The display of unassembled code looks like a listing for an assembled file.

Use: The Unassemble command operates according to the following guidelines:

- If the Unassemble command is entered without parameters, 20 hexadecimal bytes (the default) are disassembled starting at the current location to show the corresponding instructions.
- If the Unassemble command is entered with the address parameter, the default number of bytes (20 hexadecimal) is disassembled starting at the address specified. If only an offset is entered, CS is used for the segment.

- If the Unassemble command is entered with the < address> L < value> parameters, then DEBUG disassembles all bytes beginning at the address specified for the number of bytes specified by value. Entering the Unassemble command with the < address> L < value> parameters overrides the default limit (20 hexadecimal bytes).

Example: Enter:

U053A:100

DEBUG disassembles 20 hexadecimal bytes beginning at address 053A:0100:

053A:0100	206472	AND	[SI+72],AH
053A:0103	69	DB	69
053A:0104	7665	JBE	016B
053A:0106	207370	AND	[BP+DI+70],DH
053A:0109	65	DB	65
053A:010A	63	DB	63
053A:010B	69	DB	69
053A:010C	66	DB	66
053A:010D	69	DB	69
053A:010E	63	DB	63
053A:010F	61	DB	61
.			
.			
.			

If you enter:

U053A:0100 0108

The display shows:

053A:0100	206472	AND	[SI+72],AH
053A:0103	69	DB	69
053A:0104	7665	JBE	016B
053A:0106	207370	AND	[BP+DI+70],DH

However, if you enter:

U053A:100 120

or

UCS:100 L 20

Then the display appears exactly the same as above with the UCS:100 command.

If the bytes in some addresses are altered, the disassembler alters the instruction statements. The Unassemble command can be entered for the changed locations, the new instructions viewed, and the disassembled code used to edit the source file.

WRITE

Syntax: W[< address> [< drive> < sector> < sector>]]

Function: Write the data being debugged to diskette.

Use: When entered without parameters, the Write command writes a file to diskette. The file must have been named either when DEBUG was invoked or with the Name command. Both the invocation and the Name command place the file name in the standard FCB at CS:5C. The data is written starting at location CS:100 for a length specified by the BX:CX registers. 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 Write command cannot transfer absolute memory to an .EXE or .HEX file. If you are familiar with the EXE or HEX format and find it necessary to change such a file, rename it with another extension (such as .XXX) before loading it. Then DEBUG can write it back to disk.

NOTE

Do not attempt to execute a program that has been loaded in this manner, since it bypasses the relocation information in the object file.

If the Write command is entered with address the file is written beginning at that address. DEBUG always writes the number of bytes in BX:CX to the diskette file. If the Named filename exists on the diskette in the specified drive, the original file is replaced by the new data.

If the W command is entered with the drive and sector parameters, the data starting at address is written to the diskette in the drive (numeric 0=A, 1=B, and so on) beginning at the logical sector number specified by the first sector and continuing until the number of sectors specified in the second sector have been written. See the parameter description section for an explanation of sector numbering and memory limitations.

NOTE

Writing to absolute sectors is *extremely* dangerous because the process bypasses the file handler.

Example: Enter:

W

The file is written out to diskette, and the following prompt is displayed:

W
<[]

Another example:

WCS:100 1 37 2B

DEBUG writes out the contents of memory, beginning with the address CS:100, to the diskette in drive B. The data written out starts in diskette logical sector number 37 hexadecimal and consists of 2B hexadecimal sectors. When the write is complete, DEBUG displays its prompt. DEBUG reports any disk errors that may occur. Check your syntax and retry the command.

COMMAND SUMMARY

DEBUG Command	Function
C< range> < address>	Compare
D[address]	Dump
D[range]	
E< address> [list]	Enter
F< range> < list>	Fill
G[=< address> [< address> ...]]	Go
H< address> < address>	Hex
I< value>	Input
L< address> [< drive> < sector> < sector>]]	Load
M< range> < address>	Move
N< filespec> [arglist]	Name
O< value> < byte>	Output
Q	Quit
R[register-name]	Register
S< range> < list>	Search
T[=< address>][value]	Trace
U[address]	Unassemble
U[range]	
W[< address> [< drive> < sector> < sector>]]	Write

ERROR MESSAGES

During the DEBUG session, you may receive any of the following error messages. Each error terminates the DEBUG command with which it is associated, but does not terminate DEBUG itself.

Error Code	Definition
BF	Bad Flag: You attempted to alter a flag, but the pair of characters entered were an unacceptable flag value. See the Register command for the list of acceptable flag entries.
BP	Too Many Breakpoints: You specified more than ten breakpoints as parameters to the Go command. Reenter the Go command with ten or fewer breakpoints.
BR	Bad Register: You entered the Register command with an invalid register name. See the Register command for the list of valid register names.
DF	Double Flag: You entered two values for one flag. You may specify a flag value only once per RF command.

File Comparison (FILCOM)

Limitations on Source Comparisons 6-3

Invocation 6-3

Method 1 6-3

Method 2 6-4

Commands 6-4

File Specifications 6-5

Prompts 6-5

Defaults 6-7

Shortcuts 6-7

 RETURN Key 6-7

 Semicolon 6-8

Switches 6-9

Examples 6-12

The file comparison utility (FILCOM) compares the contents of two files. The differences between the two files are output to a third file. The files being compared may be either text files or binary files.

LIMITATIONS ON SOURCE COMPARISONS

When you are executing FILCOM, all available memory is used as buffer space to hold the text files. If the text files are larger than available memory, MS-DOS compares what it is able to load into the buffer space. If no matches are found within the portions of the files in the buffer space, the following message is displayed.

FILES ARE DIFFERENT

For binary files larger than available memory, both files are compared completely. The portion in memory is overlaid with the next portion from diskette. All differences are output the same as for binary comparisons of files that fit completely in memory.

INVOCATION

FILCOM can be invoked in one of two ways.

Method 1:

Enter:

FILCOM

The first prompt is then displayed. MS-DOS waits for you to enter a filename.

Source 1 Filename[.ASM]:

Method 2:

Enter:

FILCOM < source1> ,[source2], [list][/< switch> ...]

FILCOM and the filenames must be separated by commas. The slash is the only delimiter allowed between a filename and a switch letter. Switches may be placed after any of the entries in the invocation command line, but before the comma.

If you want to select the default for source 2 but not for list, enter two consecutive commas between source 1 and list entries.

For example:

FILCOM ALPHA,,GAMMA

When method 2 is used, FILCOM responds with a banner but no prompts, and performs the comparison. Method 2 permits FILCOM commands to be used in a batch file under MS-DOS, as well as permitting you to enter all commands on one line at one time. When FILCOM is finished, the operating system prompt reappears.

COMMANDS

Commands to FILCOM consist of responses to three prompts for file specifications, plus optional switches. The file specifications may be entered one at a time as the prompts appear, or all at once as part of the FILCOM invocation command (method 2).

File Specifications

All file specifications take the form:

d:filename.ext

where: d: is the letter of a diskette drive; filename is a one- to eight-character name of the file; and .ext is a one- to three-character extension to the filename. If the drive designation is omitted, FILCOM assumes the default drive. See the sections entitled “Defaults” and “Shortcuts” in this chapter for a list of the default filename extensions used with FILCOM and their effects.

Prompts

If invocation method 2 is used, no prompts are displayed. The comparison is performed, and FILCOM exits to the operating system.

If invocation method 1 is used (or else method 2 with an unacceptable filename or the name of a nonexistent file for the first source file), the first prompt is displayed.

Source 1 Filename[.ASM]:[]

Enter the name of one of the files you want compared. If the filename extension for this file is .ASM, the extension may be omitted from the entry. Otherwise, the extension must be included.

When an acceptable response has been entered, the second prompt is displayed.

Source 2 Filename[source1.BAK]:[]

Enter the name of the file you want compared to source 1. The backup file for the file named for the first prompt is the default. If your response to prompt 1 is TEST (meaning TEST.ASM), the filename TEST.BAK is displayed as the default for source 2.

If you want to compare the source 1 file with its backup file, simply press the **RETURN** key. Otherwise, enter a filename. Likewise, if the source 2 file has a filename extension of .BAK, the extension may be omitted. Otherwise, the extension must be entered also.

When an acceptable response to the second prompt has been entered, the third prompt is displayed.

```
List Filename[source1.DIF]:[]
```

Enter the name of the file to receive the list of differences. The default in this case is the name given for source 1 with a default filename extension of .DIF. Again, if the response to prompt 1 was TEST (meaning TEST.ASM), TEST.DIF is displayed as the default list filename. If this default filename is acceptable to you for the list file, simply press the **RETURN** key. Otherwise, enter the filename. (Likewise, if the filename extension .DIF is acceptable to you, the extension may be omitted, even if you specify a filename. If .DIF is an unacceptable filename extension, enter an extension along with the filename.)

When the two source files have been compared and the differences output to the list file, the operating system prompt reappears.

Defaults

FILCOM recognizes the following default extensions.

Prompt	Extension	Effect
Source 1	.ASM	Default source 1 filename extension; may be overridden.
	.OBJ	
	.EXE	
	.COM	
Source 2	.BAK	Default source 2 filename extension; may be overridden.
List	.DIF	Default list filename extension; may be overridden.

Shortcuts

FILCOM supports two shortcuts for entering commands. Both shortcuts use default responses for any prompts to which a response is not entered.

RETURN key

The source 1 filename [.ASM] prompt requires at least a filename response.

The source 2 filename [source1.BAK] and list filename [source1.DIF] prompts show a default entry—the filename entered for source 1 and a default filename extension. To select the default entry, simply press the **RETURN** key.

For example:

```
Source 1 Filename[.ASM]: TEST < RETURN>
Source 2 Filename[TEST.BAK]: < RETURN>
List Filename[TEST.DIF]: < RETURN>
```

These responses cause FILCOM to compare TEST.ASM with TEST.BAK and to output any differences to the file TEST.DIF.

You may press the **RETURN** key only for either of the prompts, regardless of what you plan to enter for the other. For example, the **RETURN** key may be pressed to select the default for source 2, but you can still specify a nondefault entry for list.

For example:

```
Source 1 Filename[.ASM]: TEST < RETURN>
Source 2 Filename[TEST.BAK]: < RETURN>
List Filename[TEST.DIF]: PAST.PRN [ ]
```

These responses cause TEST.ASM and TEST.BAK to be compared and differences to be output to the file PAST.PRN. Notice the default for source 2 was selected, but not for list.

Semicolon (;)

The semicolon (;) also selects the default responses to the source 2 and list prompts. If the semicolon is entered following the source 2 Filename prompt, the list filename prompt will not appear. That is, the semicolon selects the default response for all remaining prompts.

Unlike the **RETURN** key, once you enter semicolon, comparison begins, and you have no chance to enter another response for that comparison. Indeed, you may think of the semicolon as a message to FILCOM that you want to use all default responses.

This is especially useful when using method 2.

For example:

FILCOM TEST;

This entry causes the FILCOM banner to be displayed and then the desired comparison performed. TEST.ASM is compared with TEST.BAK and the differences are output to the file TEST.DIF. Notice this is the same as an example from the previous section illustrating the use of the **RETURN** key.

Now, consider this sample invocation:

FILCOM

Source 1 Filename[.ASM]: TEST;

This set of commands and responses produces the same result as the two previous examples. Note that you have no chance to enter alternatives to the defaults for source 2 or list when you use the semicolon.

Switches

FILCOM supports five switches. Switches are single letters appended to the method 2 invocation command line or to any of the prompt responses to control the file comparison. A switch must always be preceded by a slash.

FILCOM switches are one of two types: source comparison or binary comparison.

Source-comparison switches include:

- /A Force a source comparison of files with filename extensions .OBJ, .EXE, and .COM. Binary comparison is the default on files with these filename extensions. Files with any other filename extensions default to source comparison. Therefore, /A is not required for files that do not have one of these three filename extensions.
- /C Include comments in comparison. A comment starts with a semicolon (;), and ends with an end-of-line character. By default, comments are not included in comparisons. Thus, only functional changes in source files are detected by FILCOM. This means that comments in two files, even if different, are ignored when searching for consecutive lines that match (see the following description of the /n switch for an explanation of “match”).
- /n n is a number from one through nine; the default is three. n specifies how many consecutive lines in the two files must be the same before the two files are to match. Refer to examples 1 and 2 for a demonstration of the effects the following /n switch.

When n lines are found that match, all the lines that are different since the last n lines that matched, plus the first line of the current n lines that match are outputted. The first match line should help locate where differences occurred.

/S Include spaces and tabs in comparisons. By default, spaces and tabs are not included in comparisons. Thus, only functional changes in source files are detected. This means that spaces and tabs in the two files, even if the same, are ignored and are not used to find matches.

The binary-comparison switch is:

/B Force binary comparison of files that default to source comparison (files without the filename extensions .OBJ, .EXE, or .COM). Instead of a source comparison of lines, the two files are compared byte by byte. For differences, the offset location is output into the files and the differing bytes are output in hexadecimal. Refer to the following example 3 for a demonstration.

EXAMPLES

Example 1

Assume these two ASCII files are on diskette:

ALPHA.ASM BETA.ASM

FILE A FILE B

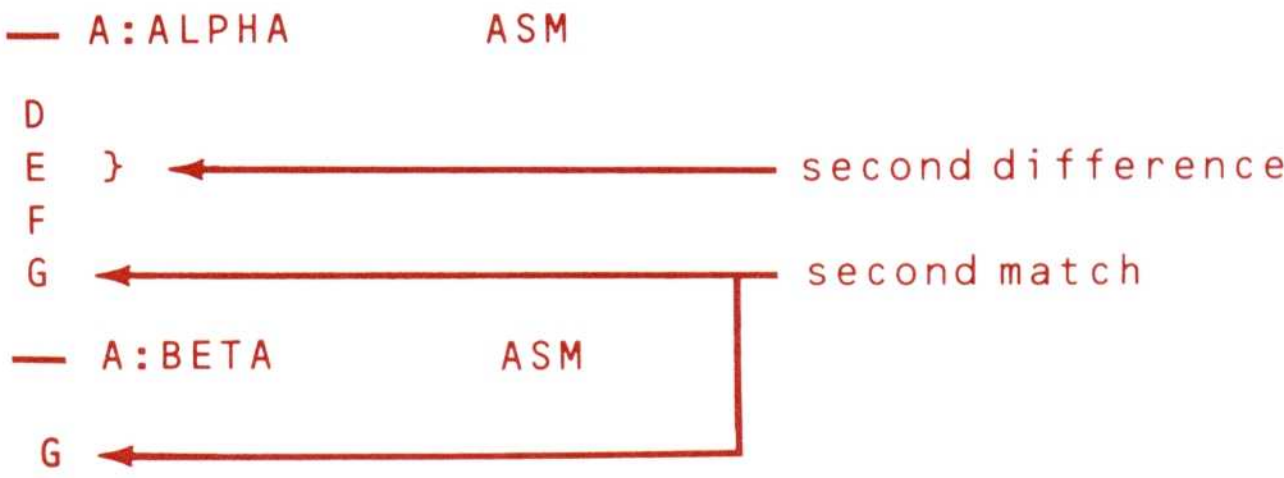
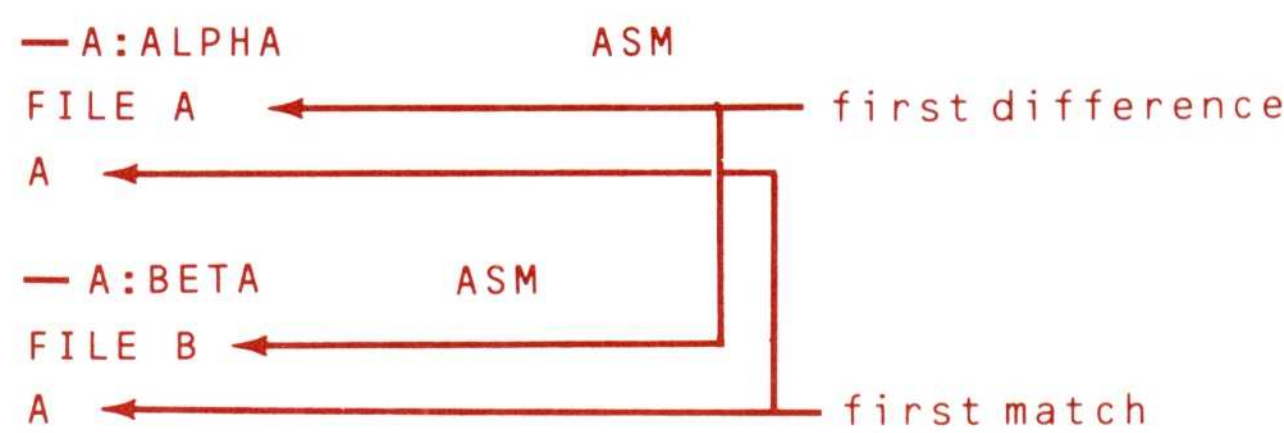
A	A
B	B
C	C
D	G
E	H
F	I
G	J
H	1
I	2
M	P
N	Q
O	R
P	S
Q	T
R	U
S	V
T	4
U	5
V	W
W	X
X	Y
Y	Z
Z	

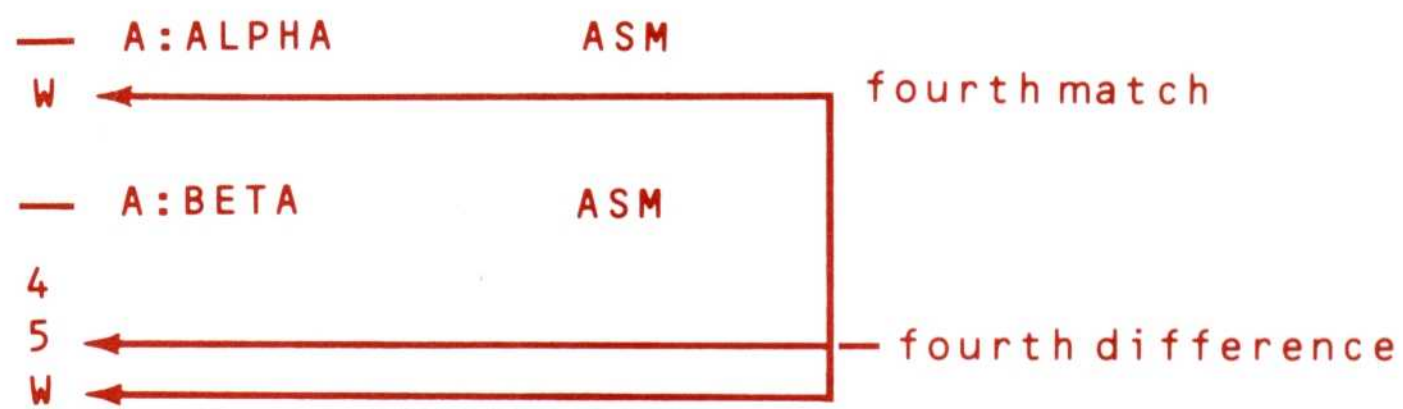
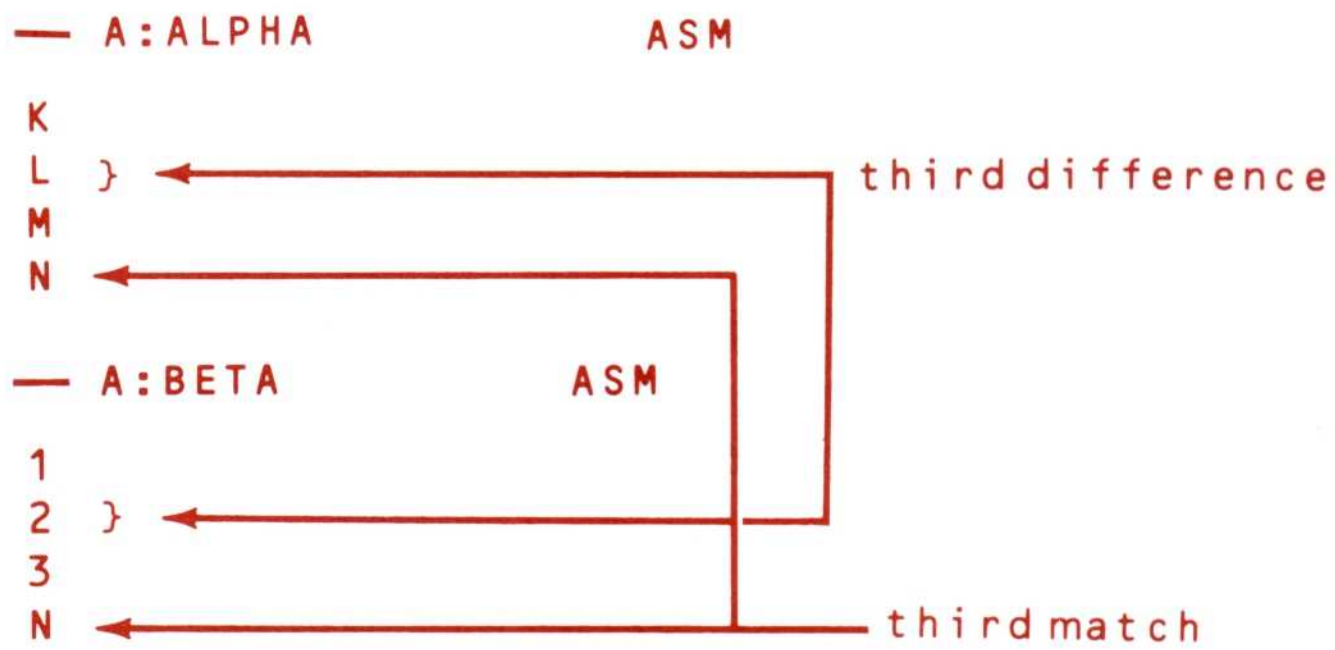
To compare the two files and output the differences on the display, enter the following method 2 command line.

FILCOM ALPHA,BETA.ASM,CON

ALPHA.ASM is compared with BETA.ASM, and the differences are output on the display. All other defaults remain intact. (When conducting a source comparison, tabs, spaces and comments do not count as matches.)

The output appears as follows on the display:





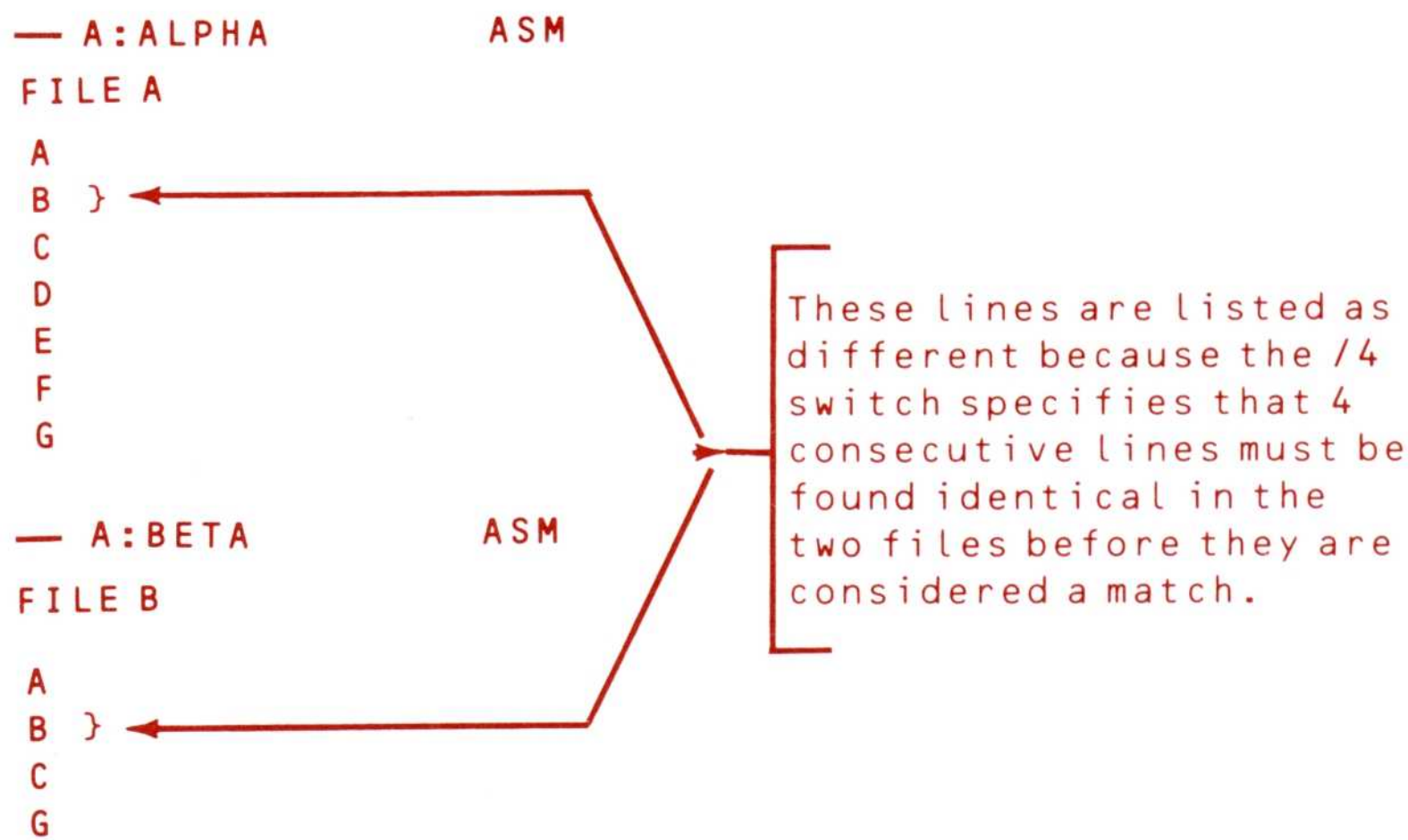
Example 2

Using the same two source files, output the differences on the printer. Also, require that four successive lines must be the same to constitute a match.

Using method 2 invocation again, enter:

FILCOM ALPHA,BETA.ASM, PRN/4

The following output should appear on the line printer.



— A:ALPHA ASM

K
L
M
N

— A:BETA ASM

1
2
3
N

— A:ALPHA ASM

W

— A:BETA ASM

4
5
W

Example 3

Using the two source files again, force a binary comparison, and then output the differences on the display.

Using method 1 invocation, enter:

```
FILCOM
```

FILCOM responds:

```
Source 1 Filename[.ASM]:
```

Entries and responses should appear as follows.

```
Source 1 Filename[.ASM]:ALPHA/B  
Source 2 Filename[ALPHA.BAK]:BETA.ASM  
List Filename[ALPHA.DIF]:CON
```

The /B switch at the end of the source 1 line forces binary comparison. This switch, and the others, may be entered at the end of any of the response entries. The switches may be, but need not be, entered on the same response line.

The display should appear as follows.

ADDRESS	ALPHA ASM	BETA ASM
00006	41	42
00012	44	47
00015	45	48
00018	46	49
0001B	47	4A
0001E	48	31
00021	49	32
00024	4A	33
00027	4B	4E
0002A	4C	4F
0002D	4D	50
00030	4E	51
00033	4F	52
00036	50	53
00039	51	54
0003C	52	55
0003F	53	56
00042	54	34
00045	55	35
00048	56	57
0004B	57	58
0004E	58	59
00051	59	5A

A

Single Diskette-Drive Users

On a single-drive system, the commands have exactly the same syntax as on a multidrive system. The difference lies in your perception of the “arrangement” of the drives. You should think of the system as having two drives (drive A and drive B). However, instead of A and B representing physical drive mechanisms of a multidrive system, the A and B designate individual diskettes.

If you specify drive B when the “drive A diskette” was last used, you are prompted to “switch drives” by swapping diskettes. If you specify drive A when the “drive B diskette” was last used, you are again prompted to change disks.

The prompts are:

```
Insert diskette for drive A:  
Strike any key when ready...
```

```
Insert diskette for drive B:  
Strike any key when ready...
```

These procedures apply to any MS-DOS command (both internal and external) which is able to access a different drive as a part of its syntax.

Also, if a command which requires access to a different drive is used in a BATCH file, the single drive procedures remain the same. When the command is executed, the system waits for you to insert the proper diskette and press a key before it continues.

Example

Assume you type the command:

A:COPY COMMAND.COM B:

and press the **RETURN** key. MS-DOS displays the prompt:

Insert diskette for drive B:
Strike any key when ready...[]

You should remove the “drive A diskette” from the drive, insert the “drive B diskette,” and strike a key.

After the file has been written to the “drive B diskette.” MS-DOS displays the normal COPY message and returns with the system prompt:

1 File(s) copied
A:[]

Remember that the letter displayed in the system prompt is the default drive. This letter does not represent the last diskette used. For example, if you were to now execute a DIRectory command with no parameters, MS-DOS would prompt you for the “drive A diskette,” since drive A is the default but the last diskette accessed was B:

A:DIR
Insert diskette for drive A:
Strike any key when ready...[]

B

File-Control-Block Definition

THE FCB

The MS-DOS file control block (FCB) is defined as follows (offsets are in decimal).

byte 0	Drive Code. 0 specifies the default drive, 1=drive A, 2=drive B, and so on. 0 is replaced by the actual drive number during an open operation.
bytes 1—8	Filename. If the file is fewer than eight characters, the name must be left justified with trailing blanks. Device names such as PRN should not include the optional colon.
bytes 9—11	Extension to filename. If fewer than three characters, must be left justified with trailing blanks. May also be all blanks.
bytes 12—13	Current block (extent). This word (low byte first) specifies the current block of 128 records, relative to the start of the file, in which sequential diskette reads and writes occur. If zero, then the first block of the file is being accessed; if one, then the second and so on. Combined with the current record field (byte 32) a particular logical record is identified. This field is set to zero by the open function call.

bytes 14–15	Record Size. Size of the record the user wishes to work with. This word may be filled immediately after an OPEN of the file if the default logical record size (128 bytes) is not desired. The Open and Create functions set this field to 128; it is also changed to 128 if a read or write is attempted with the field set to zero.
bytes 16–19	File size. This two-word field is the current size, in bytes, of the file. It may read by user programs but must not be written by them.
bytes 20–21	<p>Date. This is the date the file was created or last updated. It is set by Open to the date recorded in the diskette directory for the file. User programs may modify this field after writing to a file but before closing it to change the date recorded in the diskette directory.</p> <p>The format of this 16-bit field is as follows: bits 0–4, day of month; bits 5–8, month of year; bits 9–15, year minus 1980. All zero means no date.</p>
bytes 22–23	Time. See Date, bytes 20–21. The format is bits 0–4, seconds/2; bits 5–10, minutes; bits 11–15, hours.
bytes 24–31	Reserved for MS-DOS.
byte 32	Current record. Identifies the record (0–127) within the current block of 128 records that will be accessed with a sequential read or write function. See Current Block, bytes 12–13. The user must set this field before doing <i>sequential</i> read/write operations to the disk. This field is not initialized by the open function call.

bytes 33–36	Random Record. This field must be set if the file is to be accessed with a random read or write function. (This field is not initialized by the open function call.) If the record size is greater than or equal to 64 bytes, only the first 3 bytes are used, as a 24-bit number representing the position in the file of a record. If the record size is less than 64 bytes, all 4 bytes are used as 32-bit number for the same purpose.
-------------	--

THE EXTENDED FCB

The extended FCB is a special format used to search for files in the diskette directory with special attributes. It consists of 7 bytes in front of a normal FCB, formatted as follows:

FCB-7	Flag. FF hex is placed here to signal an extended FCB.
-------	--

FCB-6–FCB-2	Zero field.
-------------	-------------

FCB-1	Attribute byte. If bit 1=1, hidden files will be included in directory searches. If bit 2=1, system files will be included in directory searches.
-------	---

Any reference in the description of MS-DOS function calls to an FCB, whether opened or unopened, may use either a normal FCB or an extended FCB. A normal FCB has the same effect as an extended FCB with the attribute byte set to zero. If using an extended FCB, the appropriate register should be set to the first byte of the prefix instead of the drive code field.

C

Interrupts and Function Calls

INTERRUPTS

MS-DOS reserves interrupt types 20 to 3F hexadecimal for its use. This means absolute locations 80 to FF hexadecimal are the transfer address storage locations reserved by the MS-DOS. The defined interrupts are as follows with all values in hexadecimal:

- | | |
|----|--|
| 20 | Program terminate. This is the normal way to exit a program. This vector transfers to the logic in MS-DOS for restoration of the terminate and SHIFT-BRK (< CTRL-C >) exit addresses to the values they had on entry to the program. All file buffers are flushed to diskette. All files that have changed in length should have been closed (see function call 10 hexadecimal) prior to issuing this interrupt. If the changed file was not closed, its length will not be recorded correctly in the directory. When this interrupt is executed, CS <i>Must</i> contain the segment address of the program segment header. |
| 21 | Function request. See section entitled “Function Requests” in this appendix. |

22 Terminate address. The address represented by this interrupt is the address to which control transfers when the program terminates. This address is copied into the program's Program Segment Header at the time the segment is created. If a program wishes to execute a second program, it must set the terminate address prior to creation of the segment into which the program will be loaded. Otherwise, once the second program executes, its termination would cause transfer to its host's termination address. Note that MS-DOS function call 25H may be used to set this address.

23 **SHIFT-BRK** (< CTRL-C >) exit address. If the user presses and holds the **SHIFT** key and then presses the **BRK/PAUS** key during keyboard input or display output, “^C” will be displayed on the console and an interrupt type 23 hexadecimal will be executed. If the **SHIFT-BRK** routine preserves all registers, it may end with a return-from-interrupt instruction (IRET) to continue program execution. If functions 9 or 10 (buffered output and input), were being executed, then I/O will continue from the start of the line. When the interrupt occurs, all registers are set to the value they had when the original function call to MS-DOS was made. There are no restrictions on what the **SHIFT-BRK** handler is allowed to do, including MD-DOS function calls, so long as the registers are unchanged if IRET is used.

If the program creates a new segment, loads in a second program which itself changes the **SHIFT-BRK** address, the termination of the second program and return to the first will cause the **SHIFT-BRK** address to be restored to the value it had before execution of the second program.

24 Fatal error abort vector. When a fatal error occurs within MS-DOS, control will be transferred with an INT 24H. On entry to the error handler, AH will have its bit 7=0 if the error was a diskette error (probably the most common occurrence), bit 7=1 if not. If it is a diskette error, register AL contains the drive number (0=A, 1=B, and so on); AH bits 0-2 indicate the affected disk area and whether it was a read or write operation as follows:

bit 0	0 if read, 1 if write
bit 2 1	AFFECTED DISK AREA
0 0	Reserved area
0 1	File allocation table
1 0	Directory
1 1	Data area

AL, CX, DX, and DS:BX will be set up to perform a retry of the transfer with INT 25H or INT 26H (below). The lower half of the register DI will contain an error code; the upper half is undefined.

The error codes are:

0	Write protect
2	Drive not ready
4	Data error
6	Seek error
8	Sector not found
A	Disk format
C	General disk failure

The stack will be the user's stack at the time of the original MS-DOS function call and will contain the following from top to bottom:

IP	MS-DOS registers from issuing the INT 24 hexadecimal.
CS	
FLAGS	
AX	User registers at time of original MS-DOS function call (INT 21 request)
BX	
CX	
DX	
SI	
DI	
BP	
DS	
ES	
IP	From the user's original MS-DOS function call (INT 21)
CS	
FLAGS	

The registers are set so that if an IRET is executed, MS-DOS will respond according to (AL) as follows:

- (AL) = 0 ignore the error
- (AL) = 1 retry the operation (If this option is used the stack, ES, SS, SP, DS, BX, CX and DX must not be modified.)
- (AL) = 2 abort the program

Currently, the only error possible when AH bit 7=1 is a bad memory image of the file allocation table.

Before giving this routine control for disk errors, MS-DOS performs three retries. Also, this exit is taken only for errors occurring during an INT 21 function call. It is *not* used for INT 25 and INT 26 direct BIOS calls.

If you decide to handle the error yourself without returning to MS-DOS, be sure to restore your registers from the stack (above). The first and last three words shown should be removed from the stack and discarded. Also, this routine should enable interrupts because it was entered with interrupts disabled.

25

Absolute diskette read. This transfers control directly to the DOS BIOS. Upon return, the original flags are still on the stack (put there by the INT instruction). This is necessary because return information is passed back in the flags. Be sure to pop the stack to prevent uncontrolled growth. For this entry point “records” and “sectors” are the same size. The request is as follows:

(AL)	Drive number (0=A, 1=B)
(CX)	Number of sectors to read
(DX)	Beginning logical record number
(DS:BX)	Transfer address

The number of sectors specified are transferred between the given drive and the transfer address. “Logical sector numbers” are obtained by numbering each sector sequentially starting from zero and continuing across track boundaries. For example on a single-sided diskette, logical record number 0 is track 0 sector 1, whereas logical record number 12 hexadecimal is track 2 sector 3. See the parameter description section of Chapter 5, “Debugging,” for an explanation of logical sector numbering and memory limitations.

All registers but the segment registers are destroyed by this call. If the transfer was successful, the carry flag (CF) will be zero. If the transfer was not successful, CF=1 and (AL) will indicate the error as follows:

Return	Description
0	Write protect
2	Drive not ready
4	Data error
6	Seek error
8	Sector not found
A	Disk format error
C	General disk failure

- 26

Absolute diskette write. This vector is the counterpart to interrupt 25 above. Except for the fact that this is a write, the description above applies.
- 27

Terminate but stay resident. This vector is used by programs which are to remain resident when COMMAND regains control. Such a program is loaded as an executing COM file by COMMAND. After the program has initialized itself, it must set DX to its last address plus 1 in the segment in which it is executing, then execute an interrupt 27H. COMMAND will then treat the program as an extension of MS-DOS, and the program will not be overlaid when other programs are executed.

FUNCTION REQUESTS

The user requests an MS-DOS function by placing a function number in the AH register, supplying additional information in other registers as necessary for the specific function then executing an interrupt type 21 hexadecimal. When MS-DOS takes control it switches to an internal stack. User registers except AX are preserved unless information is passed back to the requester as indicated in the specific requests. The user stack needs to be sufficient to accommodate the interrupt system. It is recommended that it be 80 hexadecimal in addition to the user needs.

There is an additional mechanism provided for programs that conforms to CP/M calling conventions. The function number is placed in the CL register, other registers are set according to the function specification, and an intrasegment call is made to location 5 in the current code segment. This method is only available to functions which do not pass a parameter in AL and whose numbers are less than or equal to 24 hexadecimal. Register AX is always destroyed if this mechanism is used, otherwise it is the same as normal function requests. The functions are as follows with all values in hexadecimal.

- 0 Program terminate. The terminate and **SHIFT-BRK** exit addresses are restored to the values they had on entry to the terminating program. All file buffers are flushed, but files which have been changed in length but not closed will not be recorded properly in the diskette directory. Control transfers to the terminate address.
- 1 Keyboard input. Waits for a character to be typed at the keyboard, and then echoes the character to the display unit and returns it in AL. The character is checked for a **SHIFT-BRK**. If this key is detected an interrupt 23 hexadecimal will be executed.
- 2 Display output. The character in DL is output to the display. If a **SHIFT-BRK** is detected, after the output, an interrupt 23 hexadecimal will be executed.

-
- | | |
|---|---|
| 3 | Auxiliary input. Waits for a character from the auxiliary input device, then returns that character in AL. |
| 4 | Auxiliary output. The character in DL is output to the auxiliary device. |
| 5 | Printer output. The character in DL is output to the printer currently defined by the CONFIG command. |
| 6 | Direct console I/O. If DL is FF hexadecimal, AL returns with the zero flag clear (ZF=0) and a keyboard input character, if one is ready. If no character is waiting, the zero flag is set, and AL is 00. If DL is not FF hexadecimal, then DL is assumed to have a valid character which is output to the display unit. |
| 7 | Direct console input without echo. Waits for a character to be typed at the keyboard, then returns the character in AL. As with function 6, no checks are made on the character. |
| 8 | Console input without echo. This function is identical to function 1, except the character is not echoed. |
| 9 | Print string. On entry, DS:DX must point to a character string in memory terminated by a "\$" (24 hexadecimal). Each character in the string will be output to the display unit in the same form as function 2. |

-
- A Buffered keyboard input. On entry, DS:DX points to an input buffer. The first byte must not be zero and specifies the number of characters the buffer can hold. Characters are read from the keyboard and placed in the buffer beginning at the third byte. Reading the keyboard and filling the buffer continues until the **RETURN** key is pressed. If the buffer fills to one less than the maximum, then additional keyboard input is ignored until the **RETURN** key is pressed. The second byte of the buffer is set to the number of characters received excluding the carriage return (0D hexadecimal), which is always the last character. Editing of this buffer is described in Chapter 4.
- B Check keyboard status. If a character is available from the keyboard, AL is FF hexadecimal. Otherwise AL is 00.
- C Character input with buffer flush. First the keyboard type-ahead buffer is emptied. Then if AL is 1, 6, 7, 8, or 0A hexadecimal, the corresponding MS-DOS input function is executed. If AL is not one of these values, no further operation is done and AL returns 00.
- D Disk reset. Flushes all file buffers. Unclosed files that have been changed in size will not be properly recorded in the diskette directory until they are closed. This function need not be called before a diskette change if all files which have been written have been closed.
- E Select disk. The drive specified in DL (0=A, 1=B, and so on), is selected as the default drive. The number of drives is returned in AL.

F Open file. On entry, DS:DX points to an unopened FCB. The diskette directory is searched for the named file, and AL returns FF hexadecimal if it is not found. If it is found, AL returns 00, and the FCB is filled as follows.

If the drive code was 0 (default drive), it is changed to the actual drive used (A=1, B=2, and so on). This allows changing the default drive without interfering with subsequent operations on this file. The high byte of the current block field is set to zero. The size of the record to be worked with (FCB bytes E–F hexadecimal) is set to the system default of 80 hexadecimal. The size of the file, and the time and date are set in the FCB from information obtained from the directory.

It is the user's responsibility to set the record size (FCB bytes E–F) if the default, 80 hexadecimal, is not appropriate. It is also the user's responsibility to set the random record field and/or current block and record fields. These actions should be performed after open but before any file operations are requested.

10 Close file. This function must be called after file writes to ensure all directory information is updated. On entry, DS:DX points to an opened FCB. The diskette directory is searched and if the file is found, its position is compared with that kept in the FCB. If the file is not found in the directory, it is assumed the diskette has been changed, and AL returns FF hexadecimal. Otherwise, the directory is updated to reflect the status in the FCB, and AL returns 00.

11 Search for the first entry. On entry, DS:DX points to an unopened FCB. The diskette directory is searched for the first matching name (name could have question marks indicating any letter matched) and if none are found AL returns FF hexadecimal. Otherwise, AL returns 00, and locations at the diskette transfer address are set as follows.

1. If the FCB provided for searching was an extended FCB, then the first byte is set to FF hexadecimal, then 5 bytes of zeros, then the attribute byte from the search FCB, then the drive number used (A=1, B=2, and so on), then the 32 bytes of the directory entry. Thus the diskette transfer address contains a valid unopened extended FCB with the same search attributes as the search FCB.
2. If the FCB provided for searching was a normal FCB, then the first byte is set to the drive number used (A=1, B=2, and so on), and the next 32 bytes contain the matching directory entry. Thus the diskette transfer address contains a valid, unopened, normal FCB.

Refer to the section entitled “Diskette Directory Structure,” in this appendix for the format of directory entries.

12 Search for the next entry. After function 11 has been called and found a match, function 12 may be called to find the next match to an ambiguous request (question marks in the search filename). Both inputs and outputs are the same as function 11. The reserved area of the FCB keeps information necessary for continuing the search, so no disk operations may be performed with this FCB between a previous function 11 or 12 call and the current one.

-
- 13 Delete file. On entry, DS:DX points to an unopened FCB. All matching directory entries are deleted. If no directory entries match, AL returns FF hexadecimal, otherwise AL returns 00.
- 14 Sequential read. On entry, DS:DX points to an opened FCB. The record addressed by the current block (FCB bytes C–D hexadecimal) and the current record (FCB byte 1F hexadecimal) is loaded at the diskette transfer address, then the record address is incremented. If end of file is encountered, AL returns either 01 or 03. A return of 01 indicates no data in the record, 03 indicates a partial record is read and filled out with zeros. A return of 02 means there was not enough room in the diskette transfer segment to read one record, so the transfer was aborted. AL returns 00 if the transfer was completed successfully.
- 15 Sequential write. On entry, DS:DX points to an opened FCB. The record addressed by the current block and current record fields is written from the diskette transfer address (or, in the case of records less than sector sizes, is buffered up for an eventual write when a sector of data is accumulated). The record address is then incremented. If the diskette is full AL returns with a 01. A return of 02 means there was not enough room in the diskette transfer segment to write one record, so the transfer was aborted. AL returns 00 if the transfer was completed successfully.
- 16 Create file. On entry DS:DX points to an unopened FCB. The diskette directory is searched for an empty entry, and AL returns FF hexadecimal if none is found. Otherwise, the entry is initialized to a zero-length file, the file is opened (see function F), and AL returns 00.

-
- 17 Rename file. On entry, DS:DX points to a modified FCB which has a drive code and filename in the usual position, and a second filename starting 6 bytes after the first (DS:DX+11 hexadecimal) in what is normally a reserved area. Every matching occurrence of the first is changed to the second (with the restriction that two files cannot have the exact same name and extension). If question marks appear in the second name, then the corresponding positions in the original name will be unchanged. AL returns FF hexadecimal if no match was found, or if an attempt was made to rename to a filename that already existed, otherwise it returns 00.
- 18 Not used.
- 19 Current drive. AL returns with the code of the current default drive (0=A, 1=B, and so on).
- 1A Set diskette transfer address. The diskette transfer address is set to DS:DX. MS-DOS will not allow diskette transfers to wrap around within the segment or to overflow into the next segment.
- 1B Allocation table address. On return, DS:BX points to the allocation table for the current drive, DX has the number of allocation units, AL has the number of sectors per allocation unit, and CX has the size of the physical sector. At DS:[BX-1], the byte before the allocation table, is the dirty byte for the table. If it is set to 01, it means the table has been modified and must be written back to the diskette. If it is 00, the table is not modified. Any programs which get the address and directly modify the table must be sure to set this byte to 01 for the changes to be recorded. This byte should *never* be set to 00; instead, a DISK RESET function (0D hexadecimal) should be performed to write the table and reset the bit.
-

1C	Not used.
1D	Not used.
1E	Not used.
1F	Not used.
20	Not used.
21	Random read. On entry, DS:DX points to an opened FCB. The current block and current record fields are set to agree with the random record field, then the record addressed by these fields is loaded at the current diskette transfer address. If end of file is encountered, AL returns either 01 or 03. If 01 is returned, no more data is available. If 03 is returned, a partial record is available, filled out with zeros. A return of 02 means there was not enough room in the diskette transfer segment to read one record, so the transfer was aborted. AL returns 00 if the transfer was completed successfully.
22	Random write. On entry, DS:DX points to an opened FCB. The current block and current record fields are set to agree with the random record field, and then the record addressed by these fields is written (or, in the case of records not the same as sector sizes, buffered) from the diskette transfer address. If the diskette is full, AL returns 01. A return of 02 means there was not enough room in the diskette transfer segment to write one record, so the transfer was aborted. AL returns 00 if the transfer was completed successfully.

-
- 23 File size. On entry, DS:DX points to an unopened FCB. The diskette directory is searched for the first matching entry and if none is found, AL returns FF hexadecimal. Otherwise the random record field is set with the size of the file (in terms of the record size field rounded up), and AL returns 00. Make sure to set the record size field in the FCB before using this function call, or meaningless information will be returned.
- 24 Set random record field. On entry, DS:DX points to an opened FCB. This function sets the random record field to the same file address as the current block and record fields.
- 25 Set interrupt vector. The vector for the interrupt type specified in AL is set to the 4-byte address contained in DS:DX.
- 26 Create a new program segment. On entry, DX has a segment number at which to set up a new program segment. The entire 100 hexadecimal area at location in the current program segment (the program segment header) is copied into location zero in the new program segment. The memory size information at location 6 is updated and the current termination and **SHIFT-BRK** exit addresses are saved in the new program segment starting at 0A hexadecimal. They are restored from this area when the program terminates.

-
- 27 Random block read. On entry, DS:DX points to an opened FCB, and CX contains a record count that must not be zero. The specified number of records (in terms of the record size field) are read from the file address specified by the random record field into the diskette transfer address. If end of file is reached before all records have been read, AL returns either 01 or 03. A return of 01 indicates that end of file has been read and that the last record is complete, a 03 indicates the last record is a partial record. If wrap-around above address FFFF hexadecimal in the diskette transfer segment would occur, as many records as possible are read and AL returns 02. If all records are read successfully, AL returns 00. In any case CX returns with the actual number of records read, and the random record field and the current block and record fields are set to address the next record.
- 28 Random block write. Essentially the same as function 27 above, except for writing and a write-protect indication. If there is insufficient space on the diskette, AL returns 01, and no records are written. If CX is zero upon entry, no records are written, but the file is set to the length specified by the random record field, whether longer or shorter than the current file size (Allocation Units are released or allocated as appropriate).
- 29 Parse file name. On entry DS:SI points to a command line to parse, and ES:DI points to a portion of memory to be filled in with an unopened FCB. Leading tabs and spaces are ignored when scanning. If bit 0 of AL is equal to 1 on entry, then at most one leading filename separator will be ignored, along with any trailing tabs and spaces. The four filename separators are:

; , = +

If bit 0 of AL is equal to 1, then all parsing stops if a separator is encountered. The command line is parsed for a filename of the form d:filename.ext, and if found, a corresponding unopened FCB is created at ES:DI. The entry value of AL bits 1, 2, and 3 determine what to do if the drive, filename, or extension, respectively, are missing. In each case, if the bit is a zero and the field is not present on the command line, then the FCB is filled with a fixed value (0, meaning the default drive for the drive field; all blanks for the filename and extension fields). If the bit is a 1, and the field is not present on the command line, then that field in the destination FCB at ES:DI is left unchanged. If an asterisk (*) appears in the filename or extension, then all remaining characters in the name or extension are set to question marks (?).

The following characters are illegal within MS-DOS file specifications.

“ / [] + = ; ,

Control characters and spaces also may not be given as elements of file specifications. If any of these characters are encountered while parsing, or the period (.) or colon (:) is found in an invalid position, then parsing stops at that point.

If either a question mark or an asterisk appears in the file name or extension, then AL returns 01, otherwise 00. DS:SI will return pointing to the first character after the file name.

2A Get date. Returns date in CX:DX. CX has the year, DH has the month (1=Jan, 2=Feb, and so on), and DL has the day. If the time-of-day clock rolls over to the next day, the date will be adjusted accordingly, taking into account the number of days in each month and leap years.

- 2B Set date. On entry CX:DX must have a valid date in the same format as returned by function 2A above. If the date is indeed valid and the set operation is successful, then AL returns 00. If the date is not valid, then AL returns FF.
- 2C Get time. Returns with time of day in CX:DX. Time is actually represented as four 8-bit binary quantities, as follows: CH has the hours (0–23), CL has minutes (0–59), DH has seconds (0–59), DL has 1/100 seconds (0–99). This format is easily converted to a printable form yet can also be calculated upon (for example, subtracting two times).
- 2D Set time. On entry, CX:DX has time in the same format as returned by function 2C above. If any component of the time is not valid, the set operation is aborted, and AL returns FF hexadecimal. If the time is valid, AL returns 00.
- 2E Set/reset verify flag. On entry, DL must contain 0 and AL must contain the verify flag. The verify flag is 0 for no verify, and 1 for verify after write. If the verify flag is set, then every write operation to the diskette is immediately followed by a verify operation which rereads the information just written and compares it to the original information which remains in memory. This ensures that the write operation actually puts the intended data onto the diskette.

DISKETTE DIRECTORY STRUCTURE

Directory entries are formatted as follows:

Decimal Offset	Bytes	Description
0–7	8	Filename (E5 hexadecimal in first byte means empty directory entry, 00 means this is last entry used)

Decimal Offset	Bytes	Description
8–10	3	File name extension
11	1	Attribute—bit 1 set for hidden file, bit 2 set for system file. Both types are excluded from the normal directory search.
12–21	10	Zero field (reserved)
22–23	2	Time of creation or last update in binary fields: Bits 0–4 = secs*2 5–10 = minutes 11–15 = hours (23 is high-order byte)
24–25	2	Date of creation or last update in binary fields: Bits 0–4 = day 5–8 = month 9–15 = year (25 is high-order byte)
26-27	2	First allocation unit. First AU is always 2; for single-sided disks, this is track 0, sector 8. For double-sided, these are the two sectors starting at track 0, sector 3, side 1.
28-31	4	File size, in bytes. First word contains low-order part of size (30 bits maximum).

D

MS-DOS Messages

This appendix is provided to assist you in responding to messages that may be displayed during operation of your computer. These messages may be prompts from commands or errors from any of several sources. Each message is described and the appropriate action(s) specified.

Bold type is used to indicate the message you see on the display. Below each message is the command or condition that caused the message. (If this word is MS-DOS and/or commands, the message could have occurred from several commands or programs.)

Allocation error for file < filename >

CHKDSK.	The named file had a sector allocated to it that did not exist. The file was truncated at the end of the last valid sector.
Action:	None.

Bad command or file name

MS-DOS.	The command and/or filename entered is not valid.
Action:	Check that the command is a valid internal or external MS-DOS command; also check the diskette(s) to ensure the requested file exists.

Bad or missing Command Interpreter

STARTUP.	The diskette in drive A does not contain the correct MS-DOS files.
Action:	Try to restart the system. If this error persists make another backup from the MS-DOS diskette.

BF

DEBUG. Bad flag. An invalid flag code setting was entered.
Action: Retry the Register command with the correct code.

BP

DEBUG. Too many breakpoints. More than ten breakpoints were entered for the Go command.
Action: Reenter the Go command with ten or fewer breakpoints.

BR

DEBUG. Bad register. An invalid register name was entered.
Action: Retry the Register command with the correct name.

XXXXXXXX bytes of disk space freed

CHKDSK. Disk space shown as allocated was not actually allocated and has been freed.
Action: None.

Cannot edit .BAK file—rename file

EDLIN. .BAK files are considered to be backup files and usually should not be edited.
Action: If it is necessary to edit this file, rename it or copy it and give it a different name.

Can't create file

FILCOM. The directory already contains the maximum number of files allowed on the diskette.
Action: Use the DIR or CHKDSK command to determine the status of the diskette.

Compare error(s) on Track xx, side xx

DISKCOMP. The two diskettes contain different information.

Action: Make another backup if required.

Compare more diskettes (Y/N)?

DISKCOMP.

Action: Press **Y** if you want to compare more diskettes. Otherwise, press **N**.

Copy another (Y/N)?

DISKCOPY.

Action: Press **Y** if you want to copy another diskette. Otherwise, press **N**.

Copy complete

DISKCOPY. The destination diskette now has the same information as the source diskette.

Action: None.

Copying n side(s)

DISKCOPY. The number, n, is 1 for single-sided diskettes, and 2 for double-sided diskettes.

Action: None.

Could not read a copy of the boot sector

FORMAT. You do not have a valid MS-DOS diskette loaded in any diskette drive in the system. Your system diskette may have a bad boot sector.

Action: Insert an MS-DOS diskette in any drive, and try again.

Could not write the boot sector, disk will not boot

FORMAT. An I/O error occurred while writing the boot sector.
Action: The diskette may be useable as a data diskette.

**Data error reading drive x
Abort, Retry, Ignore?**

MS-DOS.
Action: Refer to the message “Disk error reading drive x.”

**Data error writing drive x
Abort, Retry, Ignore?**

MS-DOS.
Action: Refer to the message “Disk error reading drive x.”

Default disk not ready

CONFIG. The default diskette drive does not contain a diskette.
Action: Insert an MS-DOS system diskette into the drive and retry the command.

Default disk not a system disk

CONFIG. The diskette in the default drive is not a valid MS-DOS system diskette, and cannot be CONFIGured.
Action: Insert a valid MS-DOS system diskette into the drive and retry the command.

Default disk is write-protected

CONFIG. The MS-DOS system diskette in the default drive is write protected; the configuration information can't be written.

Action: Remove the foil tape from the write-protect notch and retry the command.

Default disk error

CONFIG. CONFIG encountered an error other than Not Ready or Write Protect when accessing the default drive.

Action: Make sure that a valid MS-DOS system diskette is in the drive.

Destination diskette is write protected Correct, then strike any key

DISKCOPY. The destination diskette is write protected.

Action: Remove the write-protect, tape and reinsert this diskette.

DF

DEBUG. Double flag. Conflicting codes were entered for a single flag.

Action: A flag value may be specified only once per RF command.

Directory error—file: < filename>

CHKDSK. No valid sectors are allocated to the named file. The filename is removed from the directory.

Action: None.

Disk boot failure

MS-DOS Start-up. An error has occurred while attempting to start MS-DOS.

Action: Try restarting. If the message repeats, use a backup MS-DOS diskette. This message will usually be followed by a system error message.

Disk error reading drive x Abort, Retry, Ignore?

MS-DOS. An error has occurred while reading from or writing to a diskette.

Action: If a diskette error occurs at any time during any diskette access, MS-DOS retries the operation three times. If the operation cannot be completed successfully, MS-DOS returns an error message in the following format:

`<type> ERROR WHILE<I/O action> ▯`

`Abort, Ignore, Retry?[]`

In this message, `< type>` may be one of the following:

WRITE PROTECT
NOT READY
SEEK
DATA
SECTOR NOT FOUND
DISK FORMAT
DISK

The `< I/O-action>` may be either of the following:

READING
WRITING

The drive < d > indicates the drive on which the error occurred. MS-DOS waits entry of one of the following responses:

A Abort. Terminate the program requesting the diskette read or write.

R Retry. Repeat the operation. This response is particularly useful if the operator has corrected the error (such as with NOT READY or WRITE PROTECT).

I Ignore. Ignore the error and continue

NOTE

Data can be *lost* when using the I response.

Usually, you will want to attempt recovery by entering responses in the order:

R (to try again)

A (to abort the program)

Disk error writing drive x Abort, Retry, Ignore?

MS-DOS.

Action: Refer to the message “Disk error reading drive x.”

Disk format error, possible hardware failure

FORMAT.

The diskette does not appear to have been formatted even though FORMAT has just finished the format operation. Probable diskette hardware failure.

Action: Report this problem to your Texas Instruments Authorized Dealer.

Disk format error reading drive x
Abort, Retry, Ignore?

MS-DOS. Diskette in question may not have been formatted prior to use.

Action: Refer to the message “Disk error reading drive x.”

Disk format error writing drive x
Abort, Retry, Ignore?

MS-DOS. Diskette in question may not have been formatted prior to use.

Action: Refer to the message “Disk error reading drive x.”

Disk full

FILCOM. There is no room to put the listing file on the diskette.

Action: Use the DIR and CHKDSK to determine the status of the diskette.

Disk full - file write not completed

EDLIN. The End command has terminated abnormally because the diskette does not have enough free space to save the entire file.

Action: Only a portion of the file will have been saved; the rest is lost. Always be sure you have sufficient free space on the diskette before beginning a edit session.

Disk unsuitable for system disk

FORMAT. Use another diskette as a system diskette.

Diskette not initialized

CHKDSK. CHKDSK could not find a recognizable directory or file allocation table on the diskette.

Action: If a DIR command shows that files exist on the diskette, you may be able to use COPY to transfer them to another diskette. In any case, use the FORMAT command to reinitialize this diskette.

Divide overflow

MS-DOS. A mathematical or logical error was encountered by the system during its execution of a program.

Action: Inspect your program for division by zero or logical errors. If this error occurs during execution of a standard utility (that is, not a user program), then report this problem to your Texas Instruments Authorized Dealer.

Drive is not a floppy disk, cannot format

FORMAT. You have attempted to format on a diskette drive not in the range A: through D:.

Action: Use only on diskette drives A: through D:.

Drive is not ready

FORMAT. There is no diskette in the drive and/or the drive door is open.

Action: Correct problem and retry.

Drive is missing or hardware failure occurred

FORMAT Cannot detect the presence of the specified drive.

Action: If specified drive is not there, don't access that drive. If it is, report this problem to your Texas Instruments Authorized Dealer.

Duplicate filename or file not found

RENAME. One of two things has happened. Either you attempted to rename a file to one that already exists on the diskette, or the file that was to be renamed couldn't be found on the specified diskette.

Action: Ensure that both filenames are valid.

Entry error

EDLIN. The last command entered contained a syntax error.

Action: Reenter the command with the correct syntax.

Entry error at or before the last character of the next line

CONFIG. There was an error in the parameter list entered on the command line. The command line is redisplayed up until the point where the error occurred.

Action: Enter the command line correctly and retry the command.

Error in EXE file

MS-DOS. An error was detected in the relocation information placed in the file by the LINK program.

Action: Relink your program.

Error in EXE/HEX file

DEBUG. The file contained invalid records or characters.

Action: Make sure that the file is a valid object file.

EXE/HEX file cannot be written

DEBUG. The absolute data in memory would require a conversion into object file format which DEBUG does not support

Action: None.

File allocation table bad, drive x Abort, Retry, Ignore?

MS-DOS.

Action: Refer to the message “Disk error reading drive x.” If you continue to get this message, use the FORMAT command to reinitialize this diskette.

File cannot be copied onto itself

COPY.

Action: Inspect the two filenames used and ensure that the copy file name is different or that it is to be copied to a different diskette.

File creation error

MS-DOS

and commands. You have attempted to add a new file to the directory. Either your diskette is full, or you have exceeded the maximum number of files allowed on the diskette.

Action: Use the DIR or CHKDSK command to determine the status of the diskette.

File not found

MS-DOS

and commands. A file that you have specified when using a command does not exist.

Action: Reenter a valid filename.

File size error for file < filename >

CHKDSK.

The size of the named file in the directory is different than its allocated size. The size in the directory is adjusted up or down (to nearest 512-byte boundary) to indicate its actual size on the diskettes.

Action: None.

Files are different

FILCOM.	The files being compared contain different information. Usually accompanied by list of differences.
Action:	None.

Files cross-linked: < filename> and < filename>

CHKDSK.	The same data block is allocated to both files.
Action:	Use the COPY command to make copies of both files; then Delete the originals. Review each file for validity and edit as necessary.

Fixups needed - base segment (hex):

EXE2BIN.	The source (.EXE) file requires a load segment.
Action	Specify the absolute segment address at which the finished load module will be loaded.

Format failure

FORMAT.	A diskette error occurred while formatting.
Action:	Try formatting a different diskette. Do not use the diskette that caused the error.

Formatting while copying

DISKCOPY.	
Action:	None.

Hardware failure occurred

FORMAT.	A disk I/O operation failed to complete.
Action:	Report this problem to your Texas Instruments Authorized Dealer.

Incompatible diskette or drive types

DISKCOMP. The source diskette and diskette drive are double sided, and either the destination diskette or the diskette drive is single sided.

Action: Ensure you are using a double-sided diskette for the destination.

Incompatible drive types

DISKCOPY. The source diskette and diskette drive are double sided, and the destination diskette drive is single sided.

Action: Ensure your diskette drives are the same.

Incompatible system size

SYS. The destination diskette contains a copy of MSDOS.SYS or IO.SYS that is smaller than the one being copied. The system transfer does not take place.

Action: Use the FORMAT/S command to put the system on a blank diskette and COPY any files to the new diskette.

Insert disk with batch file and strike any key when ready

MS-DOS. The diskette that contained the batch file being processed was removed. The batch processor is attempting to find the next command in the file.

Action: Insert the diskette in the appropriate drive and press a key.

Insert MS-DOS disk in xxxxx and strike any key when ready

MS-DOS, SYS, The MS-DOS diskette is not present in drive x.
and FORMAT.

Action: Reinsert the MS-DOS diskette in drive x.

Insert first diskette in drive x
Insert second diskette in drive y

DISKCOMP.

Action: Insert the appropriate diskette into drive x or y.

Insert source diskette in drive x
Insert destination diskette in drive y

DISKCOPY.

Action: Insert the appropriate diskette into drive x or y.

Insufficient disk space

MS-DOS and commands. You have attempted to increase the size of a file or to add a new file to the directory. Either your diskette is full, or you have exceeded the maximum number of files allowed on the diskette.

Action: Use the DIR or CHKDSK command to determine the status of the diskette.

Invalid COMMAND.COM in drive n

MS-DOS. There is a problem with a file on the MS-DOS diskette.

Action: Use a backup MS-DOS diskette.

Invalid date

DATE

Action: Enter a new date. (See Chapter 2.)

Invalid drive specification

MS-DOS and commands. An invalid drive specification was entered.

Action: Enter a valid drive specification (A:, B:, C:, and so on).

Invalid parameter

CHKDSK, DISKCOMP, DISKCOPY, FORMAT, and SYS.

Action: Enter a valid parameter.

Invalid time

TIME.

Action: Enter a valid time. (See Chapter 2.)

Larger

FILCOM. The indicated file is larger than the one to which it is being compared.

Action: None.

Line too long

EDLIN. Replace command—the string given as the replacement causes the line to expand beyond the limit of 254 characters. The command is aborted.

Action: Split the long line into two lines, the retry the operation.

Missing filename

RENAME. The new filename hasn't been specified.

Action: Specify the old and new filenames when using this command.

No room for system on destination disk

SYS. The destination diskette did not already contain the required reserved space necessary for the IO.SYS and MSDOS.SYS files.

Action: Use the FORMAT/S command to put the system onto a blank diskette, and COPY any files to the new diskette.

No room in directory for file

EDLIN. Either the directory of the specified diskette already contains the maximum number of files or an illegal diskette drive or filename was specified.

Action: Check the filename and diskette designation.

No room in disk directory

DEBUG. Either the Write command was executed without first executing a valid Name command, or the directory of the specified diskette already contains the maximum number of files.

Action: Execute a valid Name command. Use DIR and CHKDSK to examine the diskette.

Not found

EDLIN. The string specified in a Replace or Search command was not found.

Action: None.

Not ready error reading drive x Abort, Retry, Ignore?

MS-DOS.

Action: Refer to the message "Disk error reading drive x."

**Not ready error reading drive x, or
Not ready error writing drive x
Correct, then strike any key**

DISKCOMP The diskette is not inserted in the indicated
and drive.
DISKCOPY
Action: Insert the diskette in the designated drive.

**Not ready error writing drive x
Abort, Retry, Ignore?**

MS-DOS.
Action: Refer to the message “Disk error reading drive
 x.”

**Out of paper
Abort, Retry, Ignore?**

MS-DOS.
Action: Install additional paper in the printer.

**Printer fault
Abort, Retry, Ignore?**

MS-DOS.
Action: Ensure that the printer is ready.

**Printer not installed
Abort, Retry, Ignore?**

MS-DOS.
Action: Make sure the cables are connected.

**Printer not ready
Abort, Retry, Ignore?**

MS-DOS.
Action: Make sure the cables are connected and that
 the printer is online.

Program too big to fit in memory

MS-DOS and commands. There is not enough memory in the system to allow the file containing the program to be loaded.

Action: None possible, except to obtain more memory.

Sector not found error reading drive x Abort, Retry, Ignore

MS-DOS.

Action: Refer to the message "Disk error reading drive x."

Sector not found error writing drive x Abort, Retry, Ignore

MS-DOS.

Action: Refer to the message "Disk error reading drive x."

Seek error reading drive x Abort, Retry, Ignore

MS-DOS.

Action: Refer to the message "Disk error reading drive x."

Seek error writing drive x Abort, Retry, Ignore

MS-DOS.

Action: Refer to the message "Disk error reading drive x."

Sector not found error reading drive x Abort, Retry, Ignore

MS-DOS.

Action: Refer to the message "Disk error reading drive x."

**** System Error ** — xxxx**

System Operation: A system error has occurred during normal operation. A number (represented by “xxxx”) is displayed on the screen to indicate the source of the error. Some possible values are:

- | | |
|----------------|---|
| 0031 | You attempted to boot the system with no diskette in any drive. |
| Action: | Insert a diskette containing MS-DOS into one of the diskette drives and press any key to continue. |
|
 | |
| 0036 | You attempted to boot the system from a non-system diskette. |
| Action: | Insert a diskette containing MS-DOS into one of the diskette drives. Press and hold the CTRL and ALT keys and press the DEL key to reboot the system. |
|
 | |
| 1040 | An unexpected (hardware) interrupt has occurred. |
| Action: | Attempt to restart the system as described above. |
|
 | |
| 1041 | A RAM parity error has occurred. In this case you are further prompted with “Ignore, Reboot”. |
| Action: | If you wish to ignore the error and continue your work, type I , otherwise type R to reboot the system. Typing I is not recommended and may not always allow continuation. However, if the error occurred during an editing session, for example, it may be possible to continue long enough to save your file (with possible character errors, but not entire file loss). In any case, you must reboot the system as soon as possible as described above. |

1042 An unexpected interrupt has occurred.
Action: This is typically due to a programming error. Check your program for logical errors. If this error occurs during execution of a standard utility (that is, not a user program), then report this problem to Texas Instruments. Attempt to restart the system as described above.

1050 Fatal Software Error has occurred.
Action: Reboot the system as described above. Write down any related information from the display and report this error to Texas Instruments.

If you see values other than those described above, and/or encounter these errors repeatedly, refer to the “Resolving Problems” chapter in the *Texas Instruments Professional Computer Operating Instructions*.

Terminate batch job (Y/N)?

MS-DOS. This message is displayed in response to the user interrupting the processing of a batch file by pressing and holding the **SHIFT** key and then pressing the **BRK/PAUSE** key.
Action: Entering **Y** aborts the processing of the batch file. Entering **N** aborts the current command, but resumes processing the batch file with the next command.

Track 0 bad-disk unusable

FORMAT.
Action: Use another diskette.

Unexpected error, possible hardware failure

FORMAT. An error has occurred on a disk I/O which **FORMAT** never expects to happen. Probable disk hardware failure.
Action: Report this problem to your Texas Instruments Authorized Dealer.

Unrecoverable read error on drive x
Track xx, side x

DISKCOMP. Data couldn't be read from the diskette in drive x.
Action: Copy as much information as possible onto a new diskette.

Unrecoverable format error on destination

DISKCOPY. The diskette is not formattable with the /F option.
Action: The destination diskette is unusable.

Unrecoverable verify error on destination
Track xx, side x

DISKCOPY. Data is not the same on the diskettes being compared.
Action: Copy the diskette again and retry.

Unrecoverable read error on source
Track xx, side x

DISKCOPY. Data couldn't be read from the source diskette.
Action: The backup may contain incomplete information. Copy as much information as possible from the source onto a new diskette.

Unrecoverable write error on destination
Track xx, side x

DISKCOPY. Data could not be written on the destination diskette.
Action: The backup may contain incomplete information. Use another diskette as the backup.

Write protect error writing drive x
Abort, Retry, Ignore

MS-DOS.

Action:

Ensure the diskette in drive x has nothing covering the write-protect notch.

Refer to the message “Disk error reading drive x.”

Index

A

Abort D-4, D-6, D-7, D-8
AUTOEXEC.BAT 1-5, 2-24

B

Batch 2-21

C

CHKDSK 3-5
Command types 2-19
COMMAND.COM 1-6, 2-4
COMMANDS
 CHKDSK 3-5
 CONFIG 3-8
 COPY 3-12
 DATE 3-20
 DEL 3-21
 DIR 3-22
 DISKCOMP 3-23
 DISKCOPY 3-26
 ERASE 3-28
 EXE2BIN 3-29
 FORMAT 3-31
 PAUSE 3-33
 REM 3-34
 REN 3-34
 SYS 3-35
 TIME 3-36
 TYPE 3-38
 Wild-card characters (?,*) 3-21
Concatenation 3-16
Control characters 2-11
Cursor 1-7

D

Data error	D-4
DEBUG	iv, 5-1
Commands	
COMPARE	5-10
DUMP	5-10
ENTER	5-12
FILL	5-14
GO	5-15
HEX	5-16
INPUT	5-17
LOAD	5-17
MOVE	5-19
NAME	5-20
OUTPUT	5-23
QUIT	5-24
REGISTER	5-24
SEARCH	5-27
TRACE	5-28
UNASSEMBLE	5-29
WRITE	5-31
Errors	
BF - Bad flag	5-34
BP - Too many breakpoints	5-34
BR - Bad register	5-34
DF - Double flag	5-34
Flags	5-34
Default drive	1-6
DEL	3-21, 4-13
Directory	2-15, 3-22, D-5
Diskettes	1-3, 1-5
Disk errors	
Data error	D-4
Not ready error	D-17
Sector not found error	D-18
Seek error	D-18
Write fault error	D-6
Write protect error	D-6
DOWN ARROW	4-15
Drive designations	1-6
Dummy parameters (Batch)	2-23

E

EDLIN	4-1
Commands	
Append lines	4-23
Delete lines	4-23
Edit line	4-21
End editing	4-27
Insert text	4-28
List text	4-33
Quit	4-36
Replace text	4-38
Search text	4-43
Write lines	4-46
Errors	
Cannot edit .BAK file—rename file	4-47
Disk full	4-49
Entry error	4-48
Line too long	4-48
No room in directory for file	4-47
EXE files	2-13, 5-4
EXE2BIN	3-29
External commands	2-20

F

F2	4-11
F3	4-12
F4	4-14
F5	4-17
File allocation table	3-6, D-11
File control block	B-1
FORMAT	3-1

H

Hidden files	1-6
--------------------	-----

I	
INS	4-10, 4-16, 4-28
Interline Commands	4-19
Internal commands	2-20
Intraline commands	
Copy multiple characters	4-11
Copy one character	4-10
Copy template	4-12
Enter insert mode	4-16
Exit insert mode	4-16
New template	4-17
Quit input	4-15
Skip multiple characters	4-14
Skip one character	4-13
IO.SYS	1-6
L	
LINK.	2-20
M	
Memory	2-3, 2-18
MSDOS.SYS	1-5
N	
Not ready error	D-16
P	
Parameters	2-22
Prompt	1-6, 5-3
R	
RENAME (synonym for REN)	D-4, D-6, D-7, D-8, 3-34
Retry	D-4, D-6, D-7, D-8
RIGHT ARROW	4-10
S	
Sector not found error	C-6, D-18
Seek error	D-18
Start-up	1-1
Switches	3-18, 6-9
Syntax notation	1-7

T

Template 2-5

W

Wild-card characters 2-14, 3-21

THREE-MONTH LIMITED WARRANTY TEXAS INSTRUMENTS PROFESSIONAL COMPUTER SOFTWARE MEDIA

TEXAS INSTRUMENTS INCORPORATED EXTENDS THIS CONSUMER WARRANTY ONLY TO THE ORIGINAL CONSUMER PURCHASER.

WARRANTY DURATION

The media is warranted for a period of three (3) months from the date of original purchase by the consumer.

Some states do not allow the exclusion or limitation of incidental or consequential damages or limitations on how long an implied warranty lasts, so the above limitations or exclusions may not apply to you.

WARRANTY COVERAGE

This limited warranty covers the cassette or diskette ("media") on which the computer program is furnished. It does not extend to the program contained on the media or the accompanying book materials (collectively the "Program"). The media is warranted against defects in material or workmanship. **THIS WARRANTY IS VOID IF THE MEDIA HAS BEEN DAMAGED BY ACCIDENT, UNREASONABLE USE, NEGLIGENCE, IMPROPER SERVICE OR OTHER CAUSES NOT ARISING OUT OF DEFECTS IN MATERIALS OR WORKMANSHIP.**

PERFORMANCE BY TI UNDER WARRANTY

During the above three-month warranty period, defective media will be replaced when it is returned postage prepaid to a Texas Instruments Service Facility listed below or an authorized Texas Instruments Professional Computer Dealer with a copy of the purchase receipt. The replacement media will be warranted for three months from date of replacement. Other than the postage requirement (where allowed by state law), no charge will be made for the replacement. TI strongly recommends that you insure the media for value prior to mailing.

WARRANTY AND CONSEQUENTIAL DAMAGES DISCLAIMERS

ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE THREE MONTH PERIOD. TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR SPECIAL, COLLATERAL, INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES, OR DAMAGES INCURRED BY THE CONSUMER OR ANY OTHER USER ARISING OUT OF THE PURCHASE OR USE OF THE MEDIA. THESE EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED BY, COST OF REMOVAL OR REINSTALLATION, OUTSIDE COMPUTER TIME, LABOR COSTS, LOSS OF GOODWILL, LOSS OF PROFITS, LOSS OF SAVINGS, OR LOSS OF USE OR INTERRUPTION OF BUSINESS.

LEGAL REMEDIES

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

TEXAS INSTRUMENTS CONSUMER SERVICE FACILITIES

U.S. Residents:

Texas Instruments Service
Facility
P.O. Box 1444, MS 7758
Houston, Texas 77001

Canadian Residents:

Geophysical Service Inc.
41 Shelley Road
Richmond Hill, Ontario
Canada L4C 5G4

Consumers in California and Oregon may contact the following Texas Instruments offices for additional assistance or information.

Texas Instruments
Consumer Service
831 South Douglas St.
Suite 119
El Segundo, California 90245
(213) 973-2591

Texas Instruments
Consumer Service
6700 S.W. 105th
Kristin Square, Suite 110
Beaverton, Oregon 97005
(503) 643-6758

IMPORTANT NOTICE OF DISCLAIMER REGARDING THE PROGRAM

The following should be read and understood before using the software media and Program.

TI does not warrant that the Program will be free from error or will meet the specific requirements of the purchaser/user. The purchaser/user assumes complete responsibility for any decision made or actions taken based on information obtained using the Program. Any statements made concerning the utility of the Program are not to be construed as expressed or implied warranties.

TEXAS INSTRUMENTS MAKES NO WARRANTY, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE PROGRAM AND MAKES ALL PROGRAMS AVAILABLE SOLELY ON AN “AS IS” BASIS.

IN NO EVENT SHALL TEXAS INSTRUMENTS BE LIABLE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE PURCHASE OR USE OF THE PROGRAM. THESE EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED BY, COST OF REMOVAL OR REINSTALLATION, OUTSIDE COMPUTER TIME, LABOR COSTS, LOSS OF GOODWILL, LOSS OF PROFITS, LOSS OF SAVINGS, OR LOSS OF USE OR INTERRUPTION OF BUSINESS. THE SOLE AND EXCLUSIVE LIABILITY OF TEXAS INSTRUMENTS, REGARDLESS OF THE FORM OF ACTION, SHALL NOT EXCEED THE PURCHASE PRICE OF THE PROGRAM. TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR ANY CLAIM OF ANY KIND WHATSOEVER BY ANY OTHER PARTY AGAINST THE PURCHASER/USER OF THE PROGRAM.

COPYRIGHT

All Programs are copyrighted. The purchaser/user may not make unauthorized copies of the Programs for any reason. The right to make copies is subject to applicable copyright law or a Program License Agreement contained in the software package. All authorized copies must include reproduction of the copyright notice and of any proprietary rights notice.

TEXAS INSTRUMENTS PROFESSIONAL COMPUTER
MSTM-DOS Operating System
TI Part No. 2223133-0001

Original Issue: 10 December 1982

Your Name: _____

Company: _____

Telephone: _____

Department: _____

Address: _____

City/State/Zip Code: _____

Your comments and suggestions assist us in improving our products.
If your comments concern problems with this manual, please list the
page number.

Comments:

FOLD



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 6189 HOUSTON, TX

POSTAGE WILL BE PAID BY ADDRESSEE

Texas Instruments Incorporated
Attn: Marketing M/S 7896
P.O. Box 1444
Houston, TX 77001



FOLD

TEXAS INSTRUMENTS PROFESSIONAL COMPUTER
MS™-DOS Operating System
TI Part No. 2223133-0001

Original Issue: 10 December 1982

Your Name: _____

Company: _____

Telephone: _____

Department: _____

Address: _____

City/State/Zip Code: _____

Your comments and suggestions assist us in improving our products.
If your comments concern problems with this manual, please list the
page number.

Comments:

This form is not intended for use as an order blank.

FOLD



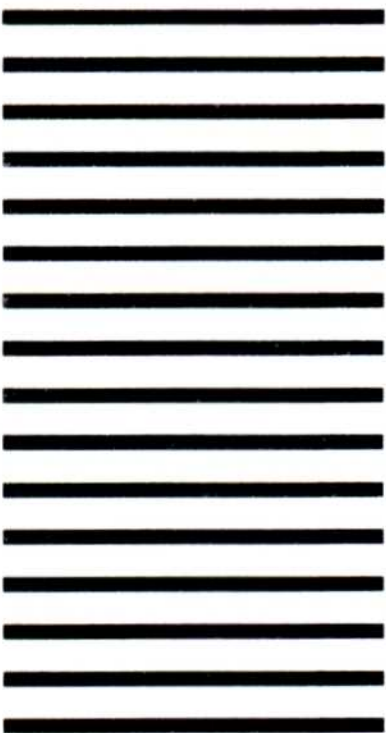
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 6189 HOUSTON, TX

POSTAGE WILL BE PAID BY ADDRESSEE

Texas Instruments Incorporated
Attn: Marketing M/S 7896
P.O. Box 1444
Houston, TX 77001



FOLD

TEXAS INSTRUMENTS PROFESSIONAL COMPUTER
MSTM-DOS Operating System
TI Part No. 2223133-0001

Original Issue: 10 December 1982

Your Name: _____

Company: _____

Telephone: _____

Department: _____

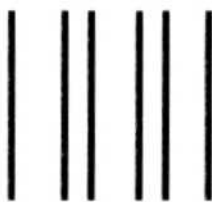
Address: _____

City/State/Zip Code: _____

Your comments and suggestions assist us in improving our products.
If your comments concern problems with this manual, please list the
page number.

Comments:

FOLD



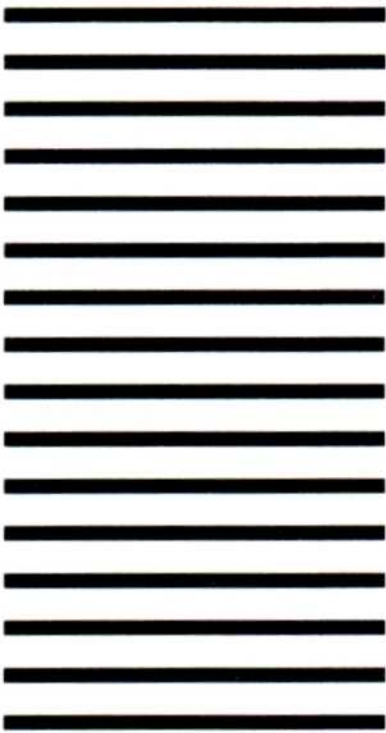
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 6189 HOUSTON, TX

POSTAGE WILL BE PAID BY ADDRESSEE

Texas Instruments Incorporated
Attn: Marketing M/S 7896
P.O. Box 1444
Houston, TX 77001



FOLD

TEXAS INSTRUMENTS

Texas Instruments reserves the right to change
its product and service offerings at any time
without notice.

Printed in U.S.A.