TeleVideo® TeleDOS User's Manual

TeleVideo

TeleDOS User's Manual

TeleVideo Part Number 125568-00 REV A

January 1984

Disclaimer

TeleVideo Systems, Inc. makes no representations or warranties with respect to this manual. Further, TeleVideo Systems, Inc. reserves the right to make changes in the specifications of the product described within this manual at any time without notice and without obligation of TeleVideo Systems, Inc. to notify any person of such revision or changes.

TeleVideo Systems, Inc.
1170 Morse Avenue
P.O. Box 3568
Sunnyvale, CA 94088
408/745-7760

TABLE OF CONTENTS

1. INTRODUCTION 1.1

What is TeleDOS 1.1
What is an Operating System 1.1
About this Manual 1.2
Syntax Notation 1.2

2. GETTING STARTED 2.1

What Happens When You First Load TeleDOS 2.1
How to Enter the Date and Time 2.1
How to Change the Default Drive 2.2
How to Format Your Diskettes 2.3
The FORMAT Command 2.3
How to Back Up Your Diskettes 2.4
The DISKCOPY Command 2.5
Automatic Program Execution 2.6
Files 2.6
How TeleDOS Keeps Track of Your Files 2.7
The DIR (Show Directory) Command 2.7
The CHKDSK (Check Disk) Command 2.8
How to Turn the System Off 2.9
Summary of Commands in this Chapter 2.9

3. MORE ABOUT FILES 3.1

Introduction 3.1
How to Name Your Files 3.1
Wildcard Characters 3.2
The ? Wildcard Character 3.2
The * Wildcard Character 3.3
Illegal Filenames 3.3
How to Copy Your Files 3.4
How to Protect Your Files 3.5
Directories 3.5
Filenames and Paths 3.8
Pathnames 3.8
Pathing and External Commands 3.9
Pathing and Internal Commands 3.10
Displaying Your Working Directory 3.11
Creating a Directory 3.12
How to Change Your Working Directory 3.12
How to Remove a Directory 3.12
Summary of Commands in this Chapter 3.13

4. LEARNING ABOUT COMMANDS 4.1

Introduction 4.1
Types of TeleDOS Commands 4.1
Command Options 4.2
Information Common to All TeleDOS Commands 4.3
Batch Processing 4.4
The AUTOEXEC.BAT File 4.6
How to Create an AUTOEXEC.BAT File 4.7
Creating a .BAT File with Replaceable Parameters . . 4.8
Executing a .BAT File 4.9
Input and Output 4.10
Redirecting Your Output 4.10
Filters 4.11
Command Piping 4.11

5. TeleDOS COMMANDS 5.1

Command Formats 5.1
TeleDOS Commands 5.2
Command Descriptions 5.4
BREAK 5.5
CHDIR (Change Directory) 5.6
CHKDSK (Check Disk) 5.7
CLS (Clear Screen) 5.10
COPY 5.11
CTTY (Change TTY) 5.14
DATE 5.15
DEL (Delete) 5.16
DIR (Directory) 5.17
DISKCOMP (Disk Compare) 5.18
DISKCOPY 5.19
EXE2BIN 5.21
EXIT 5.24
FIND 5.25
FORMAT 5.27
GRAPHICS 5.28
MKDIR (Make Directory) 5.29
MODE 5.30
MORE 5.33
PATH 5.34
PRINT 5.36
PROMPT 5.39
RECOVER 5.40
REN (Rename) 5.41
RMDIR (Remove Directory) 5.42
SET 5.43
SORT 5.45
SYS (System) 5.46
TIME 5.47
TREE 5.48
TYPE 5.50
VER (Version) 5.51

5. TeleDOS COMMANDS (Continued)

VERIFY 5.51
VOL (Volume) 5.52
Batch Processing Commands 5.53
ECHO 5.53
FOR 5.54
GOTO 5.55
IF 5.56
PAUSE 5.57
REM (Remark) 5.58
SHIFT 5.59

6. TeleDOS EDITING AND FUNCTION KEYS 6.1

Special TeleDOS Editing Keys 6.1
<COPY1> 6.3
<COPYUP> 6.4
<COPYALL> 6.5
<SKIPUP> 6.6
<NEWLINE> 6.7
<INSERT> 6.8
<SKIP1> 6.9
<VOID> 6.10

7. THE LINE EDITOR (EDLIN) 7.1

Introduction 7.1
How to Start EDLIN 7.1
Command Information 7.2
Command Options 7.4
EDLIN Commands 7.5
A (Append) 7.6
C (Copy) 7.7
D (Delete) 7.9
Edit 7.11
E (End) 7.12
I (Insert) 7.13
L (List) 7.16
M (Move) 7.18
P (Page) 7.19
Q (Quit) 7.20
R (Replace) 7.21
S (Search) 7.24
T (Transfer) 7.27
W (Write) 7.28
Error Messages 7.29

8. FILE COMPARISON UTILITY (COMP) 8.1

Introduction 8.1
Limitations on Source Comparisons 8.1
File Specifications 8.2
How to Use COMP 8.2
COMP Parameters 8.2
Difference Reporting 8.4
Redirecting COMP Output to a File 8.4
Examples 8.5
Error Messages 8.8

9. THE LINKER PROGRAM 9.1

Introduction 9.1
Overview of the Linker Program 9.1
Definitions You Need to Know 9.3
Files That the Linker Uses 9.4
Input File Extensions 9.5
Output File Extensions 9.5
VM.TMP (Temporary) File 9.5
How to Start the Linker 9.6
Summary of Starting Methods 9.6
Method 1: Prompts 9.6
Method 2: Command Line 9.7
Method 3: Response File 9.8
Command Characters 9.9
Command Prompts 9.10
Linker Parameters 9.12
Sample Link Session 9.14
Error Messages 9.16

10. DEBUG UTILITY 10.1

Overview of DEBUG 10.1
How to Start DEBUG 10.1
Method 1: DEBUG 10.1
Method 2: Command Line 10.2
Parameters 10.3
Commands 10.5
A (Assemble) 10.7
C (Compare) 10.9
D (Dump) 10.10
E (Enter) 10.11
F (Fill) 10.13
G (Go) 10.14
H (Hex) 10.16
I (Input) 10.17
L (Load) 10.18
M (Move) 10.20
N (Name) 10.21
O (Output) 10.23
Q (Quit) 10.23

10. DEBUG UTILITY (Continued)

R (Register) 10.24
S (Search) 10.27
T (Trace) 10.28
U (Unassemble) 10.29
W (Write) 10.30
Error Messages 10.32

APPENDICES

A. Instructions for Users With Single-Drive Systems . . A.1
B. Disk Errors B.2
C. ANSI Escape Sequences C.4
D. How to Configure Your System D.10
E. TeleDOS Diskette File List E.12

TABLES

1-1 Syntax Notation 1.2
2-1 Chapter 2 Commands 2.9
3-1 Sample Directory Names 3.8
3-2 Internal Commands 3.10
3-3 Chapter 3 Commands 3.13
4-1 Chapter 4 Commands 4.12
5-1 TeleDOS Commands 5.2
5-2 Batch Command Summary 5.4
6-1 Special Editing Functions 6.2
7-1 EDLIN Commands 7.4
9-1 Linker Command Prompts 9.7
9-2 Linker Command Characters 9.9
9-3 Linker Parameters 9.12
10-1 DEBUG Command Parameters 10.3
10-2 DEBUG Commands 10.6
10-3 DEBUG Error Codes 10.32
C-1 Graphic Function Parameters C.7
C-2 Screen Parameters C.8

FIGURES

1-1 Hardware/Software Relationships 1.1
2-1 DISKCOPY Command 2.6
3-1 Copying Files to Another Disk 3.4
3-2 Copying Files on the Same Disk 3.5
3-3 Sample Hierarchical Directory Structure 3.7
4-1 TeleDOS Batch File Steps 4.5
4-2 How TeleDOS Uses the AUTOEXEC.BAT File 4.7
6-1 Text Line and Template 6.1
9-1 Linking Operation 9.2
9-2 How Memory is Divided 9.3

1. INTRODUCTION

WHAT IS TeleDOS?

TeleDOS is a disk operating system based on Microsoft's MS-DOS. Through TeleDOS you communicate with the computer, disk drives, and printer, managing these resources to your advantage.

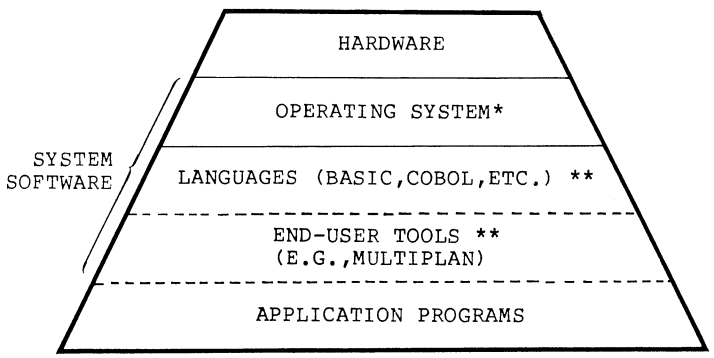
TeleDOS allows you to create and keep track of files, run and link programs, and access peripheral devices (for example, printers and disk drives) that are attached to your computer. TeleDOS is an important advance in microprocessor operating systems.

WHAT IS AN OPERATING SYSTEM?

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

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

Figure 1-1
Hardware/Software Relationships



* MUST ADAPT TO NEW HARDWARE
** IF ADAPTED TO OPERATING SYSTEM, THESE DON'T CHANGE

ABOUT THIS MANUAL

This manual describes TeleDOS and tells you how to use it. This chapter introduces some basic TeleDOS concepts.

Chapter 2 discusses how to start using TeleDOS and how to format and back up your disks.

Chapter 3 tells you about files--what they are and how to use them.

Chapters 4 through 6 introduce TeleDOS commands and Chapter 7 describes the line editor, EDLIN. Read these chapters carefully, they contain information on protecting your data, system commands, and the TeleDOS editing commands.

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

If you are writing programs and want to link separately-produced object modules and create relocatable modules, Chapter 9 describes the linker utility.

Chapter 10 describes how to use the DEBUG program for testing binary and executable object files.

Appendices to this manual include instructions for users with computers that have only one disk drive.

SYNTAX NOTATION

Table 1-1 lists the notation used throughout this manual in descriptions of command and statement syntax:

Table 1-1
Syntax Notation

Symbol	Description
[ ]	Square brackets indicate that the enclosed entry is optional.
< >	Angle brackets indicate data you enter. When the angle brackets enclose lower-case text, type in an entry defined by the text; for example, <filename>. When the angle brackets enclose upper-case text (or a capital followed by lower-case), you must press the key named by the text; for example, <Ctrl>.
<CR>	Indicates you are to press the <Enter> key. CR stands for Carriage Return.

Table 1-1 (Continued)
Syntax Notation

Symbol	Description
/	This symbol, when used between two or more keys, indicates you are to press the keys simultaneously. For example, <Ctrl>/<Break> indicates you are to hold down the <Ctrl> key and press the <Break> key, then release both keys.
^	The caret is used to indicate the control key. It is followed by an upper-case letter, indicating you are to hold down the <Ctrl> key and press that letter key. As an example, if you see <^C>, hold down the <Ctrl> key and press the <C> key.
{ }	Braces indicate that you have a choice between two or more entries. At least one of the entries enclosed in braces must be chosen unless the entries are also enclosed in square brackets. The choices are separated by the bar (|) symbol.
|	A bar indicates an OR statement in a command. When used with a TeleDOS filter, the bar indicates a pipe (see Chapter 4 for a description of filters and pipes).
...	Ellipses indicate that an entry may be repeated as many times as needed or desired.
CAPS	Capital letters indicate portions of statements or commands that must be entered, exactly as shown.

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

Bold face is used in examples to show the items you would enter at the keyboard.

As an example, the format of the directory command is:

DIR [<path>][<filename>[.<ext>]][/P][/W]

To look at the directory information for file MYFILE.BAS in your working directory on the default drive, you would enter:

DIR MYFILE.BAS<CR>

The DIR listed in caps in the command format must be entered. Since MYFILE.BAS resides in your working directory, a path does not need to be included. MYFILE is the <filename> and .BAS is the file <ext>. Neither the /P or /W options were requested.

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

2. GETTING STARTED

WHAT HAPPENS WHEN YOU FIRST LOAD TeleDOS?

Follow the instructions in your user's manual to insert and load the TeleDOS diskette into your computer. Loading TeleDOS takes from 3 to 45 seconds, depending on the size of memory in your computer.

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

When the command processor is loaded, the following date prompt is displayed on your screen (the underscore represents the cursor):

```
Current date is Tue 1-01-1980
Enter new date: _
```

You must now enter the current date and time.

HOW TO ENTER THE DATE AND TIME

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

- mm is a one- or two-digit number from 1-12 representing the current month
- dd is a one- or two-digit number from 1-31 representing the day of month
- yy is a two-digit number from 80-99 (the 19 is assumed), or a four-digit number from 1980-2099 (representing year)

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

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

NOTE! If you make a mistake while typing, press the the <Backspace> key to erase one letter at a time.

If you enter an invalid date or form of date, TeleDOS again prompts you with:

```
Enter new date: _
```

After you respond to the new date prompt and enter your answer by pressing <CR>, the following time prompt is displayed:

```
Current time is  0:0:14.32
Enter new time: _
```

Enter the current time in the hh:mm[ss.ss] format, where:

hh is a one- or two-digit number from 0-23 representing hours (12 midnight would be entered as 0, 1:00 pm would be entered as 13)

mm is a one- or two-digit number from 0-59 representing the current minute

ss.ss is a number from 00.00 to 59.99 representing the current second. This entry is optional.

TeleDOS uses this time value to keep track of when you last updated and/or created files on the system. Notice that TeleDOS uses a 24-hour clock.

Example:

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

You should only use the colon (:) to separate hours and minutes. If you enter an invalid number separator, TeleDOS repeats the prompt.

NOTE! A <CR> can be entered in response to the date and time prompts to accept the default values listed in the prompts. TeleDOS uses the current date and time as directory entries for every file you create or update.

You have now completed the steps for starting TeleDOS.

HOW TO CHANGE THE DEFAULT DRIVE

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

```
TeleVideo Personal Computer DOS Vers. 2.11
(c) Copyright TeleVideo Systems, Inc. 1983
(c) Copyright Microsoft Corp. 1981, 1982, 1983

A>_
```

The A> is the TeleDOS **system prompt** from the command processor. It tells you that TeleDOS is ready to accept commands. The A character in the system prompt represents the default disk drive. This means that TeleDOS searches only drive A for any filenames you may enter, and writes files to that drive unless you specify a different drive.

You can ask TeleDOS to search drive B by changing the drive designation or by specifying the drive designation B: with the filename. To change the disk drive designation, enter the new drive letter followed by a colon. For example:

A>	(TeleDOS system prompt)
A>**B:<CR>**	(you enter B: in response to the prompt)
B>	(system responds with B>, and drive B is now the default drive)

The system prompt B> appears and TeleDOS searches only drive B until you specify a different default drive.

NOTE! If you have only one disk drive in your computer, turn to Appendix A, Instructions for Users with Single-Drive Systems, for important information.

HOW TO FORMAT YOUR DISKETTES

You must format all new diskettes before they can be used by TeleDOS.

A blank diskette must be formatted with the TeleDOS FORMAT command. The FORMAT command changes the diskette to a format that TeleDOS can use; it also analyzes the diskette for defective tracks. If the diskette already contains data, formatting the diskette will erase the data. Although this can be a convenient way to make a blank disk, it is recommended that the TeleDOS DELete command be used for this purpose. Refer to Chapter 5 for more information on the DELete command.

The FORMAT Command

The syntax of the FORMAT command is:

FORMAT [<d>:]

where:

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

Note that the brackets identify optional information. If you do not specify a disk drive (for example, A: or B:), TeleDOS formats the diskette in the default drive.

With the TeleDOS diskette already in drive A, you are ready to format your new diskette. The following command formats the new diskette in drive B.

FORMAT B:<CR>

NOTE! TeleDOS commands can be entered in upper-case letters, lower-case letters, or a combination of both.

TeleDOS displays the following message:

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

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

```
Formatting...
```

while the FORMAT program is formatting your diskette.

When the formatting is finished, TeleDOS displays a message similar to this:

```
Formatting...Format complete

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

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

Enter Y to format another diskette. Enter N to end the FORMAT program. You do not need to press <CR> after entering Y or N.

Refer to Chapter 5, TeleDOS Commands, for more information on the FORMAT command.

HOW TO BACK UP YOUR DISKETTES

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

The DISKCOPY Command

The DISKCOPY command copies the contents of a diskette onto another diskette. You can use this command to duplicate the TeleDOS diskette, or a diskette that contains your own files or programs. DISKCOPY is the fastest way of copying a diskette, because it copies the entire diskette in one operation, including TeleDOS system files if they exist.

The format of the DISKCOPY command is:

DISKCOPY [<d>:] [<d>:]

The first d is the disk drive that contains the diskette that you want to copy (source); the second d is the disk drive that contains the blank or target diskette.

For example, if you want to make a copy of your TeleDOS diskette which is in drive A, enter:

DISKCOPY A: B:<CR>

TeleDOS responds:

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

Make sure the TeleDOS diskette is in drive A and insert a blank, diskette in drive B. Press any character key after inserting the diskettes and TeleDOS begins copying the TeleDOS diskette. TeleDOS displays:

COPYING 9 SECTORS PER TRACK 2 SIDE(S)

If the blank diskette was not formatted, TeleDOS displays:

Formatting while copying.

After TeleDOS has copied the diskette, TeleDOS displays:

Copy completed

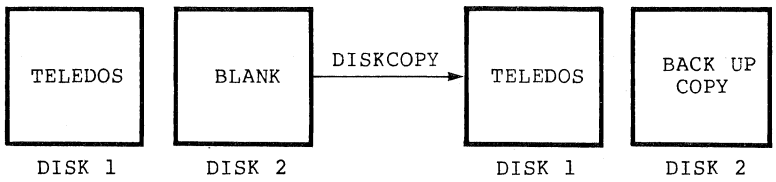
Copy Another (Y/N)?

Enter Y (for Yes) if you wish to copy another diskette with DISKCOPY. If you enter N (for No), the default drive prompt is displayed. You do not need to press <CR> after entering Y or N.

You now have a duplicate copy of your TeleDOS diskette in drive B. This duplicate copy should be used for everyday use. The original or master TeleDOS diskette should be stored in a safe place and used only for making working copies.

Figure 2-1 illustrates the DISKCOPY command.

Figure 2-1
DISKCOPY Command



Diskettes must be the same size and density to be copied with the DISKCOPY command. Refer to Chapter 5, TeleDOS Commands, for more information on the DISKCOPY command.

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

AUTOMATIC PROGRAM EXECUTION

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

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

FILES

What is A File?

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

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

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

How TeleDOS Keeps Track Of Your Files

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

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

These two system areas, the directories and the File Allocation Table, enable TeleDOS to recognize and organize the files on your disks. The File Allocation Table is copied onto a new diskette when you format it with the TeleDOS FORMAT command and one empty directory is created, called the **root directory** (see Chapter 3 for a description of the root directory).

The DIR (Show Directory) Command

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

DIR A:<CR>

TeleDOS responds with a directory listing of all the files in the current directory on your TeleDOS diskette. The display should look similar to this:

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

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

      xx File(s)        xxxxx bytes free
```

NOTE! Two TeleDOS system files, IO.SYS and MSDOS.SYS, are **hidden files** and will not appear when you issue the DIR command.

This listing includes the filenames, file extensions, the size in bytes, and the date and time these files were created or last modified.

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

DIR MYFILE.TXT<CR>

displays all of the directory information (name of file, size of file, date last edited) for the file MYFILE.TXT.

For more information on the DIR command, refer to Chapter 5, TeleDOS Commands.

The CHKDSK (Check Disk) Command

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

To check the disk in drive A, enter:

CHKDSK A:<CR>

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

HOW TO TURN THE SYSTEM OFF

There is no logoff command in TeleDOS. To end your terminal session, open the disk drive doors and remove the diskettes. Then, turn off your computer.

NOTE! **Always remove your diskettes from the disk drives before you turn off your computer.**

SUMMARY OF COMMANDS IN THIS CHAPTER

Table 2-1
Chapter 2 Commands

Command	Purpose	Format
FORMAT	Formats diskettes	FORMAT [<d>:]
DISKCOPY	Copies diskettes	DISKCOPY [<d>:] [<d>:]
DIR	Lists directory information	DIR [<d>:] [<filename>]
CHKDSK	Checks for errors on disk	CHKDSK [<d>:]

3. MORE ABOUT FILES

INTRODUCTION

In Chapter 2, you learned that directories contain the names of your files. This chapter discusses how to name and copy your files. It also contains more information about the TeleDOS hierarchical directory structure, which makes it easy for you to organize and locate your files.

HOW TO NAME YOUR FILES

The name of a typical TeleDOS file looks like this:

NEWFILE.EXE

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

A filename can be from one to eight characters long. The filename extension can be three or fewer characters. You can enter filenames in lower- or upper-case letters. TeleDOS translates lower-case letters into upper-case letters.

In addition to the filename and the filename extension, a drive designation may be included. A drive designation tells TeleDOS to look on that drive to find the entered filename. For example, to find directory information about the file NEWFILE.EXE which is located on drive A (and drive A is NOT the default drive), enter the following command:

DIR A:NEWFILE.EXE<CR>

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

If drive A is the default drive, TeleDOS searches only drive A for the filename NEWFILE.EXE and the drive designation is not necessary. A drive designation is needed if you want TeleDOS to look on a drive other than the current default drive.

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

A-Z 0-9 $ & #

% ' ( ) - @

^ { } ~ ` ! _

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

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

NOTE! Spaces are not allowed anywhere within a file specification.

Remember that square brackets indicate optional items. Angle brackets (<>) mean that you supply the text for the item. Note that the drive designation (<d>:) is not required if the file is on the default drive. You do not have to give your filename a filename extension.

Examples of file specifications are:

B:MYPROG.COB
A:YOURPROG.EXT
A:NEWFILE
TEXT

WILDCARD CHARACTERS

Two special characters (called wildcard characters) can be used within filenames and extensions in TeleDOS commands: the asterisk (*) and the question mark (?). These special characters give you greater flexibility when using filenames in TeleDOS commands.

The ? Wildcard Character

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

DIR TEST?RUN.EXE<CR>

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

TEST1RUN.EXE
TEST2RUN.EXE
TEST6RUN.EXE

The * Wildcard Character

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

DIR TEST*.EXE<CR>

lists all directory entries on the default drive with filenames that begin with the characters TEST and have an extension of the above DIR command:

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

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

Examples:

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

DIR A:NEWFILE.*<CR>

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

DIR B:*.TXT<CR>

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

ILLEGAL FILENAMES

TeleDOS treats some device names specially, and certain three-letter names are reserved for the names of these devices. These three-letter names cannot be used as filenames or extensions, but are used in commands to represent a device.

AUX — Used when referring to input from or output to an auxiliary device (such as a printer or disk drive).

CON — Used when referring to keyboard input or to output to the terminal console (screen).

LST or
PRN Used when referring to the printer device.

NUL Used when you do not want to create a particular file, but the command requires an input or output filename.

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

HOW TO COPY YOUR FILES

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

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

The format of the COPY command is:

COPY <filespec> [<filespec>]

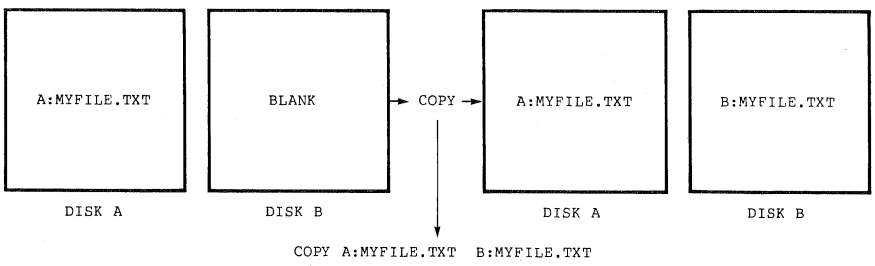
For example,

COPY A:MYFILE.TXT B:MYFILE.TXT<CR>

copies the file MYFILE.TXT on drive A to a file named MYFILE.TXT on drive B. A duplicate copy of MYFILE.TXT now exists.

Figure 3-1 illustrates how to copy files to another disk.

Figure 3-1
Copying Files to Another Disk

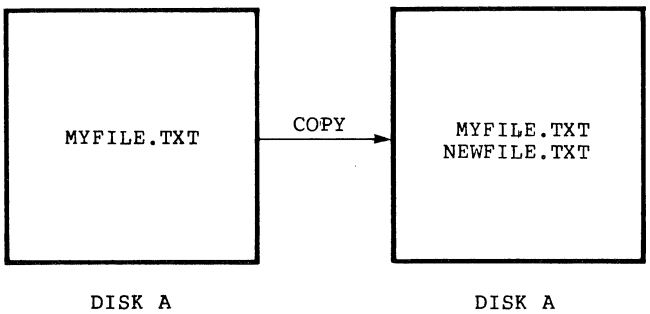


If you want to duplicate the file named MYFILE.TXT on the same drive, enter:

COPY A:MYFILE.TXT A:NEWNAME.TXT<CR>

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

Figure 3-2
Copying Files on the Same Disk



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

HOW TO PROTECT YOUR FILES

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

DIRECTORIES

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

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

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

A hierarchical directory structure can be thought of as a tree structure. Directories are branches of the tree and files are the leaves, except that the tree grows downward; that is, the root is at the top. The root directory is the first level in the directory structure. It is the directory that is automatically created when you format a disk and start putting files in it. You can create additional directories and subdirectories by following the instructions in Chapter 4, Learning About Commands.

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

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

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

Figure 3-3 illustrates a typical hierarchical directory structure.

Figure 3-3
Sample Hierarchical Directory Structure

```
                    ROOT
                     |
        ------------------------------------
        |      |      |          |          |
      GAMES   BIN    USER     ACCOUNTS   PROGRAMS
                     |
              -----------------
              |      |          |
             JOE    SUE        MARY
                     |          |
              ----------     TEXT.TXT
              |         |
          TEXT.TXT    FORMS
```

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

A directory of games, named GAMES

A directory of all external commands, named BIN.

A USER directory containing separate subdirectories for all users of the system

A directory containing accounting information, named ACCOUNTS

A directory of programs, named PROGRAMS

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

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

DIR \USER\SUE\FORMS<CR>

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

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

DIR \USER\MARY<CR>

FILENAMES AND PATHS

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

Pathnames

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

The complete format for a pathname is:

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

where [<d>:][\<directory>][\<directory>]... is known as a **path.**

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

The following directory listings in Table 3-1 refer to Figure 3-3.

Table 3-1
Sample Directory Names

Symbol	Explanation
\	Indicates the root directory.
\PROGRAMS	A subdirectory under the root directory.
\USER\MARY\TEXT.TXT	A typical full pathname. TEXT.TXT is a file in directory MARY, which is a subdirectory of directory USER.

USER\SUE	A path refering to directory SUE in subdirectory USER of the working directory. If the working directory is the root (\), it names \USER\SUE.
TEXT.TXT	Name of a file in the working directory MARY or SUE.

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

- . TeleDOS uses this shorthand notation to indicate the name of the working directory in all hierarchical directory listings. TeleDOS automatically creates this entry when a directory is made.

- .. This shorthand notation is used to indicate the working directory's parent directory. If you enter:

  DIR ..<CR>

  TeleDOS lists the files in the parent directory of your working directory.

  If you enter:

  DIR ..\..<CR>

  TeleDOS lists the files in the parent's PARENT directory.

Pathing and External Commands

External commands reside on disks as program files. They must be loaded into memory from the disk before they start to run. (For more information on external commands, refer to Chapter 4, Learning About Commands.)

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

For example, if you are in a working directory named \BIN\PROG, and all TeleDOS external commands are in \BIN, you must tell TeleDOS to choose the \BIN path to find the FORMAT command. The command

PATH \BIN<CR>

tells TeleDOS to search in your working directory and the \BIN directory for all commands. You must specify this path each time TeleDOS is loaded from disk. TeleDOS now searchs in \BIN for the external commands. If you want to know what the current path is, enter only the word PATH and the current value of PATH is displayed.

For more information on the TeleDOS command PATH, refer to Chapter 5, TeleDOS Commands.

Pathing and Internal Commands

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

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

The format of these four commands is shown below.

Table 3-2
Internal Commands

Command	Notes
COPY <pathname> <pathname> COPY <pathname> <path>	If the second entry is a path, the file is copied into that directory.
DEL <pathname> DEL <path>	If a path is entered, all the files in that directory are deleted. **NOTE!** The prompt "Are you sure (Y/N)?" will be displayed if you try to delete a path. Press Y to complete the command, or N to abort.
DIR <pathname> DIR <path>	Displays the directory for a specific path or pathname.
TYPE <pathname>	TeleDOS displays the file on your screen in response to the TYPE pathname command.

Displaying Your Working Directory

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

CHDIR<CR>

TeleDOS displays:

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

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

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

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

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

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

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

Creating A Directory

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

MKDIR NEWDIR<CR>

After this command has been executed, a new directory exists in your tree structure under your working directory. You can also make directories anywhere in the tree structure by entering MKDIR and a path. TeleDOS automatically creates the . and .. entries in the new directory.

To put files in the new directory, use the TeleDOS line editor, EDLIN. Chapter 7 describes how to use EDLIN to create and save files.

How To Change Your Working Directory

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

CHDIR \USER<CR>

changes the working directory from \USER\JOE to \USER. You can specify any path after the command to travel to different branches and leaves of the directory tree.

The command:

CHDIR ..<CR>

puts you in the parent directory of your working directory.

How to Remove a Directory

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

RMDIR NEWDIR<CR>

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

RMDIR \BIN\USER\JOE<CR>

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

DEL \BIN\USER\SUE<CR>

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

SUMMARY OF COMMANDS IN THIS CHAPTER

Table 3-3
Chapter 3 Commands

Command	Purpose	Format
COPY	Copies files	COPY <pathname> [[<path>]<filename>[.<ext>]]
PATH	Sets TeleDOS search path	PATH [<path>]
CHDIR	Displays working directory; changes directories	CHDIR [<path>]
MKDIR	Makes a new directory	MKDIR <path>
RMDIR	Removes a directory	RMDIR <path>

4. LEARNING ABOUT COMMANDS

INTRODUCTION

Commands are a way of communicating with the computer. By entering TeleDOS commands at your keyboard, you can ask the computer to perform useful tasks. There are TeleDOS commands that:

Compare, copy, display, delete, and rename files

Copy and format disks

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

Analyze and list directories

Enter date, time, and remarks

Copy TeleDOS system files to another disk

TYPES OF TeleDOS COMMANDS

There are two types of TeleDOS commands:

Internal commands

External commands

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

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

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

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

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

CHKDSK	FORMAT	RECOVER
DISKCOMP	GRAPHICS	SORT
DISKCOPY	MODE	SYS
EXE2BIN	MORE	TREE
FIND	PRINT	

COMMAND OPTIONS

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

The format of all TeleDOS commands is:

```
Command [options...]
```

where the following notation is used to describe the options:

d: Refers to a disk drive designation.

filename Refers to any valid name for a disk file. The filename option does not refer to a device or to a disk drive designation.

.ext Refers to an optional filename extension consisting of a period and 1-3 characters. When used, filename extensions immediately follow filenames.

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

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

path Refers to a path to follow to reach the required directory. A path is entered in the following format:

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

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

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

parameters — Parameters are options that control TeleDOS commands. They are preceded by a forward slash (for example, /P).

arguments — Provide more information to TeleDOS commands. You usually choose between arguments; for example, ON or OFF.

INFORMATION COMMON TO ALL TeleDOS COMMANDS

The following information applies to all TeleDOS commands:

1. Commands are usually followed by one or more options.

2. Commands and options may be entered in upper-case, lower-case, or a combination of both.

3. Commands and options must be separated by delimiters. Because they are easiest, you will usually use the space and comma as delimiters. For example:

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

   You can also use the semicolon (;), the equal sign (=), or the tab key as a delimiter in TeleDOS commands.

   NOTE! In this manual, a space is used as the delimiter in commands.

4. Do not place delimiters within a file specification. The colon and the period already serve as delimiters.

5. When instructions say, "Press any key", you can press any alphanumeric or punctuation key.

6. You must include the filename extension when referring to a file that already has a filename extension.

7. You can abort commands when they are running by pressing <Ctrl>/<Break>.

8. Commands take effect only after you have pressed the <CR> key.

9. Wildcard characters (global filename characters) and device names (for example, PRN or CON) are not allowed in the names of any commands. You may only use them in command options.

10. When commands produce a large amount of output on the screen, the display automatically scrolls to the next screen. You can press <Ctrl>/<Num Lock> to suspend the display. Press any key to resume the display on the screen.

11. TeleDOS editing and function keys can be used when entering commands. Refer to Chapter 6, TeleDOS Editing and Function Keys, for a complete description of these keys.

12. The prompt from the command processor is the default drive designation plus a greater-than sign; for example, A>.

13. Disk drives are referred to as source drives and destination (or target) drives. A source drive is the drive you will be transferring information from. A destination drive is the drive you will be transferring information to.

BATCH PROCESSING

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

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

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

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

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

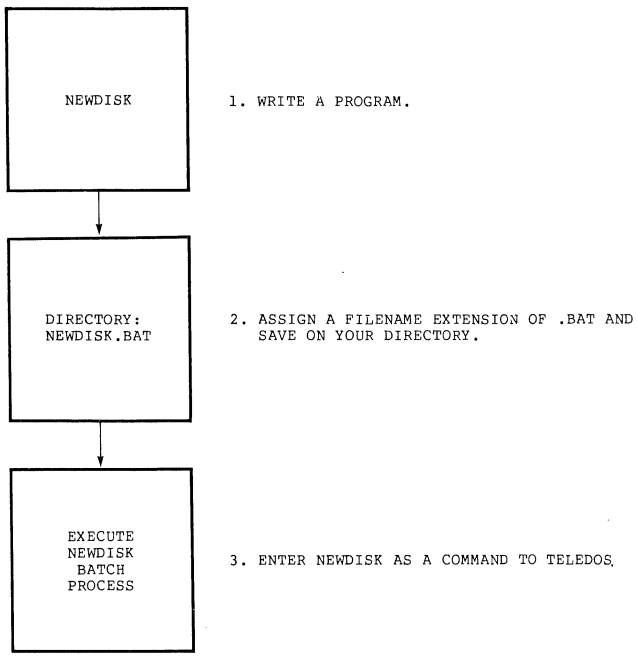
To execute this .BAT file, enter the filename without the .BAT extension:

NEWDISK<CR>

The result is the same as if each of the lines in the .BAT file had been entered at the keyboard as individual commands.

Figure 4-1 illustrates the three steps used to write, save, and execute a TeleDOS batch file.

Figure 4-1
TeleDOS Batch File Steps



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

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

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

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

4. If you press <Ctrl>/<Break> while in the batch mode, this prompt appears:

   Terminate batch job (Y/N)?

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

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

5. If you remove the disk containing a batch file being executed, TeleDOS prompts you to insert it again before the next command can be read.

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

THE AUTOEXEC.BAT FILE

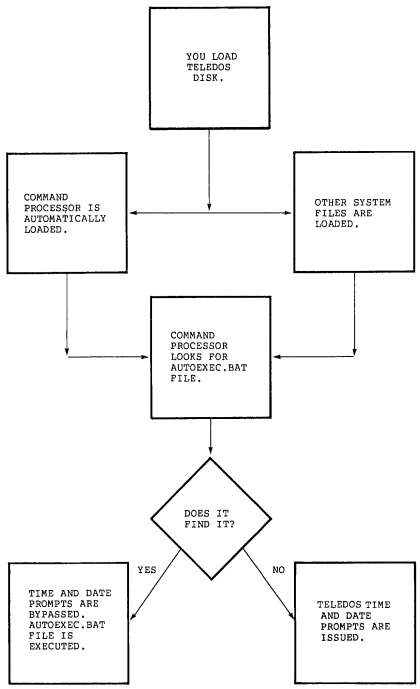
As discussed in Chapter 2, an AUTOEXEC.BAT file allows you to automatically execute programs when you start TeleDOS. Automatic Program Execution is useful when you want to run a specific package under TeleDOS and when you want TeleDOS to execute a batch program automatically each time you start the system.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

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

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

If TeleDOS does not find an AUTOEXEC.BAT file when you first load the TeleDOS disk, then the date and time prompts are displayed. Figure 4-2 illustrates how TeleDOS uses the AUTOEXEC.BAT file.

Figure 4-2
How TeleDOS Uses the AUTOEXEC.BAT File



How To Create An AUTOEXEC.BAT File

If, for example, you want to automatically load BASIC and run a program called MENU each time you start TeleDOS, you could create an AUTOEXEC.BAT file as follows:

1. Enter: **COPY CON: AUTOEXEC.BAT<CR>**

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

2. Now enter: **BASIC MENU<CR>**

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

3. Press the <F6> key (inserts a control Z, end-of-file character); then press <CR> to put the command BASIC MENU in the AUTOEXEC.BAT file. TeleDOS responds with:

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

4. The MENU program will now run automatically whenever you start TeleDOS.

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

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

CREATING A .BAT FILE WITH REPLACEABLE PARAMETERS

There may be times when you want to create an applications program and run it with different sets of data. The data may be stored in various TeleDOS files.

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

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

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

Now, press **<F6>** and **<CR>**. TeleDOS responds with the message:

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

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

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

NOTES

1. Up to 10 dummy parameters (%0-%9) can be specified. Refer to the TeleDOS command SHIFT in Chapter 5 if you wish to specify more than 10 parameters.

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

Executing A .BAT File

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

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

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

To execute the MYFILE batch process, enter:

MYFILE A:PROG1 B:PROG2<CR>

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

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

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

The following illustrates how TeleDOS replaces each of the above parameters:

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

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

INPUT AND OUTPUT

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

Redirecting Your Output

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

DIR<CR>

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

DIR >MYFILES<CR>

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

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

DIR >>MYFILES<CR>

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

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

SORT <NAMES >LIST1<CR>

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

Filters

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

TeleDOS filters include FIND, MORE, and SORT. Their functions are described below:

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

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

Command Piping

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

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

DIR | SORT<CR>

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

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

DIR | SORT >DIREC.FIL<CR>

TeleDOS creates a file named DIREC.FIL on your default drive. DIREC.FIL contains a sorted listing of the directory on the default drive, since no other drive was specified in the command. To specify a drive other than the default drive, enter:

DIR | SORT >B:DIREC.FIL<CR>

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

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

DIR | SORT | MORE<CR>

sorts your directory, shows it to you one screen at a time, and puts --MORE-- at the bottom of your screen when there is more output to be seen.

You will find many uses for piping commands and filters.

Table 4-1
Chapter 4 Commands

Command	Purpose	Syntax
REM	Adds comment line for batch files	REM [remark]
PAUSE	Suspends execution of a batch file	PAUSE [comment]
FIND	Searches for string of text	FIND "<string>" [<filename>]
MORE	Pages through a file 23 lines at a time	MORE
SORT	Sorts text	SORT

5. TeleDOS COMMANDS

COMMAND FORMATS

The following notation indicates how you should format TeleDOS commands:

1. You must enter any words shown in capital letters. These words are called keywords and must be entered exactly as shown. You can enter these keywords in any combination of upper- and lower-case; TeleDOS converts all keywords to upper-case.

2. You supply the text for any items enclosed in angle brackets (< >). For example, you should enter the name of your file when <filename> is shown.

3. Items in square brackets ([ ]) are optional. If you wish to include optional information, do not include the square brackets, only the information within the brackets.

4. An ellipsis (...) indicates that you may repeat an item as many times as you want.

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

6. The notation <filename> implies:

   <filename>

7. The notation <filespec> implies:

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

8. The notation <path> implies:

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

9. The notation <pathname> implies

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

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

TeleDOS COMMANDS

The following TeleDOS commands are described in this chapter. Note that synonyms for commands are enclosed in parentheses.

Table 5-1
TeleDOS Commands

Command	Description
BREAK	Sets the Control Break check
CHDIR	Changes directories; prints working directory (CD)
CHKDSK	Scans the directory of the default or designated drive and checks for consistency
CLS	Clears screen
COPY	Copies file(s) specified
CTTY	Changes console TTY
DATE	Displays and sets date
DEL	Deletes file(s) specified (ERASE)
DIR	Lists requested directory entries
DISKCOMP	Compares two diskettes
DISKCOPY	Copies disks
EXE2BIN	Converts executable files to binary format
EXIT	Exits command and returns to lower level
FIND	Searches for a constant string of text
FORMAT	Formats a disk to receive TeleDOS files
GRAPHICS	Prints the contents of a graphics screen display on a graphics printer
MKDIR	Makes a directory (MD)

MODE	Sets mode of operation for the screen, the printer, and modem ports
MORE	Displays output one screen at a time
PATH	Sets a command search path
PRINT	Background print feature
PROMPT	Designates system prompt
RECOVER	Recovers a bad disk
REN	Renames first file as second file (RENAME)
RMDIR	Removes a directory (RD)
SET	Sets one string value to another
SORT	Sorts data alphabetically, forward or backward
SYS	Transfers TeleDOS system files from the default drive to the drive specified
TREE	Displays all of the directory paths found on the specified drive
TIME	Displays and sets time
TYPE	Displays the contents of file specified
VER	Prints TeleDOS version number
VERIFY	Verifies writes to disk
VOL	Prints the volume identification label

Table 5-2
Batch Command Summary (Command Extensions)

Command	Description
ECHO	Turns batch file echo feature on/off
FOR	Batch command extension
GOTO	Batch command extension
IF	Batch command extension
PAUSE	Pauses for input in a batch file
REM	Displays a comment in a batch file
SHIFT	Increases number of replaceable parameters in batch process

COMMAND DESCRIPTIONS

The remainder of this chapter provides a detailed description of the TeleDOS commands. The commands are described using the following format.

COMMAND NAME

Type	States whether the command is an internal or external command (see Chapter 4).
Synonym	An alternate command name that can be used for the same command function.
Purpose	Gives a brief description of the command function.
Format	Lists the format for entering the command.
Comments	Detailed information on how to use the command and examples.
Notes	Additional command information.

BREAK (Control Break) Command

Type Internal

Purpose Sets the TeleDOS Control Break check; displays the current BREAK state.

Format BREAK [{ON|OFF}]

Comments This command allows you to instruct TeleDOS when to check for the <Ctrl>/<Break> function. When ON, TeleDOS checks for a control break whenever a program requests any TeleDOS function.

If BREAK is OFF, TeleDOS only checks for the <Ctrl>/<Break> function when requested to do screen, keyboard, printer, or auxiliary device functions. OFF is the default state when TeleDOS is started.

If BREAK is entered without an ON or OFF option, the current BREAK state is displayed.

CHDIR (Change Directory) Command

Type Internal

Synonym CD

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

Format CHDIR [<path>]

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

CHDIR \BIN\USER\JOE\FORMS<CR>

TeleDOS puts you in the new directory.

A shorthand notation is also available with this command:

CHDIR ..<CR>

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

CHDIR used without a path displays your working directory. If your working directory is \BIN\USER\JOE on drive B, and you enter:

CHDIR<CR>

TeleDOS displays:

B: \BIN\USER\JOE

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

CHKDSK (Check Disk) Command

Type External

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

Format CHKDSK [<d>:][<filename>][/F][/V]

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

A sample status report follows:

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

 65536 bytes total memory
 53152 bytes free
```

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

To redirect the CHKDSK output to a file, enter:

CHKDSK A:><filename><CR>

The errors are sent to the specified file. Do not use the /F parameter if you redirect the CHKDSK output.

The following errors are corrected automatically if you specify the /F parameter:

Invalid drive specification

Invalid parameter

Invalid sub-directory entry

Cannot CHDIR to <filename>
Tree past this point not processed

First cluster number is invalid
entry truncated

Allocation error, size adjusted

CHKDSK (Check Disk) Command

Has invalid cluster, file truncated

Disk error reading FAT

Disk error writing FAT

<filename> contains non-contiguous blocks

All specified file(s) are contiguous

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

Incorrect DOS version

You cannot run CHKDSK on versions of DOS that are not 2.0 or higher.

Insufficient memory
Processing cannot continue

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

Errors found, F parameter not specified
Corrections will not be written to disk

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

Invalid current directory
Processing cannot continue

Restart the system and re-run CHKDSK.

Cannot CHDIR to root
Processing cannot continue

The disk you are checking is bad. Try restarting TeleDOS and RECOVER the disk.

<filename> is cross linked on cluster

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

CHKDSK (Check Disk) Command

Lost clusters found in y chains
Convert lost chains to files (Y/N)?

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

CHKDSK then displays:

X bytes disk space freed

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

X bytes disk space would be freed

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

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

Insufficient room in root directory
Erase files in root and repeat CHKDSK

CHKDSK cannot process until you delete files in the root directory.

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

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

If a <filename> is included in the CHKDSK command, CHKDSK displays the number of non-contiguous areas occupied by the specified file.

CLS (Clear Screen) Command

Type	Internal
Purpose	Clears the terminal screen.
Format	CLS
Comments	The CLS command causes TeleDOS to send the ANSI escape sequence ESC[2J which clears your display screen.

COPY Command

Type Internal

Purpose Copies one or more files to another disk. This command can also copy files on the same disk.

Format COPY <pathname> [[<path>]<filename>[.<ext>]][/V]

Comments If a second file is not given, the copy will be on the default drive and will have the same name as the original file (the first file listed). If the first file is on the default drive and a second file is not specified, the COPY is aborted. (Copying files to themselves is not allowed.) TeleDOS displays the error message:

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

The second file listed may take the following forms:

1. A drive designation (d:) only - the original file is copied with the original filename to the designated drive.

2. A path (directory) only - the original file is copied into that directory with the original filename.

3. A filename only - the original file is copied to a file on the default drive with the new filename.

4. A filespec - the original file is copied to a file on the designated drive with the specified filename.

5. A full pathname - the original file is copied to a file in the listed directory on the specified drive with the specified filename.

The /V parameter requests TeleDOS to verify that the sectors written on the destination disk are recorded properly. Although there are rarely recording errors when you run COPY, you can verify that critical data has been correctly recorded. This option results in the COPY command running more slowly as TeleDOS checks each entry recorded on the disk.

COPY Command

Reserved device names can also be used with the COPY command. For example:

```
COPY CON: <filename>
COPY CON: AUX:
COPY CON: LPT1
COPY <filename> CON:
COPY <filename> AUX:
COPY AUX: CON:
```

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

For example,

COPY A.TXT + B.TXT + B:C.TXT BIGFILE.CRP<CR>

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

To combine several files using wildcard characters into one file, you could enter:

COPY *.LST COMBIN.LRG<CR>

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

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

COPY *.LST + *.REF *.SAV<CR>

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

COPY *.LST + *.REF COMBIN.SAV<CR>

COPY Command

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

COPY *.LST ALL.LST<CR>

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

COPY compares the filename of the input file with the filename of the destination. If they are the same, that one input file is skipped, and the error message, "Content of destination lost before copy", is printed. Further concatenation proceeds normally.

If the COPY command is used to concatenate files without specifying a destination file, the additional files are appended to the end of the first file. For example:

COPY ALL.LST + *.LST<CR>

appends all *.LST files, except ALL.LST itself, to ALL.LST. This command will not produce an error message and is the correct way to append files using the COPY command.

CTTY (Change TTY) Command

Type Internal

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

Format CTTY <device>

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

CTTY AUX<CR>

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

CTTY CON<CR>

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

DATE Command

Type Internal

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

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

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

Comments If you enter DATE, DATE responds with the message:

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

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

You can also enter a particular date after the word DATE on the command line, as in:

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

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

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

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

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

If the options or separators are not valid, DATE displays the message:

```
Invalid date
Enter new date: _
```

DATE then waits for you to enter a valid date.

DEL (Delete) Command

Type Internal

Synonym ERASE

Purpose Deletes all files with the designated filespec.

Format DEL [<path>][<filename>[.<ext>]]

Comments The global wildcard characters ? and * can be used in the filename and extension.

To delete all the files in the current directory, enter:

DEL [<d>:]*.*<CR>

TeleDOS responds with the following verification question:

Are you sure (Y/N)?

Enter Y or y and <CR> to delete all the files in the current directory. Enter N or n and <CR> to abort.

To delete all the files in a specific directory, enter:

DEL <path><CR>

The . and .. file entries in a sub-directory cannot be deleted.

The system files MSDOS.SYS and IO.SYS cannot be deleted.

DIR (Directory) Command

Type Internal

Format DIR [<path>][<filename[.<ext>]][/P][/W]

Purpose Lists the files in a directory.

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

The wildcard characters ? and * (question mark and asterisk) may be used in the filename option. Note that for your convenience, the DIR commands in Table 5-3 are equivalent.

Table 5-3
DIR Command Equivalents

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

Two parameters may be specified with DIR. The /P parameter selects Page Mode. With /P, display of the directory pauses after the screen is filled. To display the next screen of directory listings, press any key.

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

DISKCOMP (Disk Compare) Command

Type External

Purpose Compares the contents of two diskettes. Normally used after using DISKCOPY to ensure the two diskettes are identical.

Format DISKCOMP [<d>:] [<d>:][/1][/8]

Comments DISKCOMP is used to compare two diskettes. DISKCOMP compares all 40 tracks on a track-for-track basis and displays a message if the tracks are not equal. The message lists the track and side where the mismatch is found. Upon completion, the following prompt is displayed:

Compare more diskettes (Y/N)?

Press Y to do another comparison on the same drives (a prompt is displayed to insert the diskettes). Press N to abort the DISKCOMP program.

The /1 parameter forces DISKCOMP to compare only side one of the diskettes.

The /8 parameter forces DISKCOMP to compare only 8 sectors per track.

If both drive options are omitted, a single-drive comparison is performed on the default drive.

If the second drive is omitted, the default drive is used as the secondary drive.

DISKCOMP determines the number of sides and sectors per track to be compared based on the first diskette read (the diskette in the first drive designation parameter entered).

NOTE! If a backup diskette is created using the COPY command, the backup will contain the same information but in different locations. The DISKCOMP command would indicate that these two diskettes are different.

DISKCOPY Command

Type External

Purpose Copies the contents of the diskette in the source drive to the diskette in the destination drive.

Format DISKCOPY [<d>:] [<d>:] [/1]

Comments The first option you specify is the source drive. The second option is the destination (target) drive.

If the diskette in the destination drive is not formatted, DISKCOPY formats the diskette while copying.

The /1 parameter forces DISKCOPY to copy only side one of the source diskette.

You can specify the same drives or you may specify different drives. If the drives designated are the same, a single-drive copy operation is performed.

If the command:

DISKCOPY A: B:<CR>

is given, TeleDOS displays the following message:

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

DISKCOPY waits for you to press any key before continuing. The following is displayed while the copying is in progress:

```
COPYING n SECTORS PER TRACK m SIDE(S)
```

where n is either 8 or 9, and m is either 1 or 2. If the target diskette needs formatting, the following additional message is displayed:

```
Formatting while copying.
```

After copying, DISKCOPY prompts:

```
Copy completed

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

If you press Y, the insert prompt is displayed for a copy using the same source and destination drives.

DISKCOPY Command

To end the DISKCOPY program, press N.

A <CR> is not required after entering Y or N.

Notes

1. If the target diskette does not have the same format as the source diskette, DISKCOPY will format the diskette while it is copying.

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

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

4. On a single-drive system, all prompts are for drive A regardless of the drive designations you enter.

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

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

   For example:

   COPY A:*.* B:<CR>

   copies all files from the diskette in drive A to the diskette in drive B.

6. DISKCOPY determines the number of sides and the sectors per track to copy, based on the source drive and diskette.

7. If disk errors are encountered on either diskette, DISKCOPY indicates the drive, track, and side where the error occurred, and then continues with the copy. In this case, the target diskette may or may not be usable, depending on the validity of the data being copied to the bad sector(s).

EXE2BIN Command

Type External

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

Format EXE2BIN <pathname> [<pathname>]

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

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

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

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

EXE2BIN Command

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

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

File cannot be converted

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

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

File not found

The file is not on the disk specified.

Insufficient memory

There is not enough memory to run EXE2BIN.

File creation error

EXE2BIN cannot create the output file. Run CHKDSK to determine if the directory is full, or if some other condition caused the error.

Insufficient disk space

There is not enough disk space to create a new file.

EXE2BIN Command

Fixups needed - base segment (hex):

The source (.EXE) file contained information indicating that a load segment is required for the file. Specify the absolute segment address at which the finished module is to be located.

File cannot be converted

The input file is not in the correct format.

WARNING -Read error on EXE file.

Amount read less than size in header
This is a warning message only.

EXIT Command

Type Internal

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

Format EXIT

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

COMMAND<CR>

You can now enter the DIR command and TeleDOS displays the directory for the default disk. When you enter EXIT, you return to the previous level (your applications program).

FIND Command

Type External

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

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

Comments FIND is a filter that takes as arguments a string and a series of filenames. FIND displays all the lines that contain the specified string from the listed files.

If no files are specified, FIND takes the input on the screen and displays all the lines that contain the specified string.

Parameters for FIND are:

/V FIND displays all the lines not containing the specified string.

/C FIND displays only the number of lines that contained a match in each file.

/N FIND displays the line number in addition to the line of text containing the match.

The string should be enclosed in quotes. Example:

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

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

The command

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

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

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

Incorrect DOS version

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

FIND
Command

FIND: Invalid number of parameters

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

FIND: Syntax error

You entered an illegal string when issuing the FIND command.

FIND: File not found <filename>

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

FIND: Read error in <filename>

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

FIND: Invalid parameter <option-name>

You specified an option that does not exist.

FORMAT Command

Type External

Purpose Formats the disk in the specified drive to accept TeleDOS files.

Format FORMAT [<d>:][/S][/1][/8][/V]

Comments This command initializes the directory and file allocation tables and marks any defective tracks as reserved to prevent them from being used. If no drive is specified, the disk in the default drive is formatted.

The /S parameter requests the FORMAT program to also copy the operating system files to the new diskette. The files are copied in the following order:

IO.SYS
MSDOS.SYS
COMMAND.COM

The /1 parameter forces the target diskette to be formatted for single-sided disk drives. The default format is for double-sided drives.

The /8 parameter formats the diskette with 8 sectors per track. The default format is 9 sectors per track.

The /V parameter requests FORMAT to prompt for a volume label after the disk is formatted. The volume label consists of 1 to 11 characters. The volume label can be used for keeping track of diskettes.

The FORMAT program displays a status report indicating the total diskette space, space marked as defective, space currently allocated to files (with /S), and size of space available for user's files.

NOTE! The FORMAT program destroys any previously-existing data on the diskette.

GRAPHICS Command

Type External

Purpose Prints the contents of a graphics display screen on a graphics printer.

Format GRAPHICS

Comments Press the <Shift> and <PrtSc> keys simultaneously to print the contents of the screen on the printer. If the screen is in the text mode, the display is printed in under 30 seconds. If the display is in the graphics mode, each time the <Shift>/<PrtSc> keys are pressed, the following occurs:

1. In the 320 x 200 color graphics mode, the display is printed in up to four shades of gray.

2. In the 640 x 200 color graphics mode, the display image is printed sideways on the paper. The upper-right corner of the display is printed on the upper left-corner of the paper.

To invoke the print feature from a program, use the following coding example:

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

MKDIR (Make Directory) Command

Type Internal

Synonym MD

Purpose Creates a sub-directory on the specified disk.

Format MKDIR <path>

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

MKDIR \USER<CR>

creates a subdirectory \USER in your root directory.

To create a directory named JOE under \USER, enter:

MKDIR \USER\JOE<CR>

MODE Command

Type External

Purpose Sets the mode of operation for the printer and the modem ports.

Format

```
MODE LPT#:[<n>][,[<m>][,P]]
```

or

```
MODE n
```

or

```
MODE [<n>],<m>[,T]
```

or

```
MODE COM<n>:<baud>[,parity[,databits[,stopbits[,P]]]]
```

or

```
MODE LPT#:=COM<n>
```

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

Option 1 (For the printer)

```
MODE LPT#:[<n>][,[<m>][,P]]
```

where:

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

MODE Command

Option 2 (For setting the display mode)

MODE <n> or

MODE [<n>],<m>[,T]

where:

- n — is 40, 80, BW40, BW80, CO40, CO80, or MONO
  - 40 — sets the display width to 40 characters per line
  - 80 or MONO — sets the display width to 80 characters per line
  - BW40 — sets the display mode to Black and White with 40 characters per line
  - BW80 — sets the display mode to Black and White with 80 characters per line
  - CO40 — sets the display mode to color with 40 characters per line
  - CO80 — sets the display mode to color with 80 characters per line
- m — is R or L for shift right or left
- T — requests a test pattern used to align the display

The display can be shifted for readability. In the 40 column width, the display is shifted one character at a time, and two characters at a time in the 80 column width. If the T option is included, a prompt asks if the screen is aligned properly. If you enter Y, MODE ends. If you enter N, the shift is repeated.

MODE Command

Option 3 (For the RS-232C port)

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

where:	n	1 or 2 for the serial port number
	baud	110, 150, 300, 600, 1200, 2400, 4800, or 9600 (only the first two characters are required; subsequent characters are ignored)
	parity	N for none, O for odd, or E for even (default = E)
	databits	7 or 8 (default = 7)
	stopbits	1 or 2 (default = 1 if baud <> 110 = 2 if baud = 110)
	P	the serial port is being used for a serial interface printer

The protocol parameters are used to initialize the RS-232C serial port. A minimum entry must include the baud rate.

Option 4 (To redirect the parallel printer output to the serial port)

MODE LPT#:=COM<n>

where:	#	1, 2, or 3 for the printer number
	n	1 or 2 for serial port number

The output for printer LPT# is redirected to serial port n. The serial port must first be initialized using the option 3 format of the MODE command before using this format.

To redirect the printer output back to the parallel port, use the MODE LPT#:n,m format command.

MORE Command

Type External

Purpose Sends output to the screen one screen at a time.

Format MORE

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

Pressing a character key displays another screen of information. This process continues until all the input data has been output to the screen.

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

TYPE MYFILES.COM | MORE<CR>

TeleDOS displays the file MYFILES.COM one screen at a time.

PATH (Set Search Directory) Command

Type Internal

Purpose Sets a command path to be searched for external commands or batch files that were not found by a search of the current directory.

Format PATH [<path>][;[<path>]...]

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

To tell TeleDOS to search your \BIN\USER\JOE directory for external commands, enter:

PATH \BIN\USER\JOE<CR>

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

You can tell TeleDOS to search more than one path by specifying several paths separated by semicolons. For example,

PATH \BIN\USER\JOE;\BIN\USER\SUE;\BIN\DEV<CR>

tells TeleDOS to search the directories specified by the above paths to find external commands. TeleDOS searches the paths in the order specified in the PATH command.

The PATH command with no options prints the current path. If you enter:

PATH ;<CR>

TeleDOS sets the NUL path, meaning that only the working directory is searched for external commands.

PATH Command

As an example, assume the program SPECIAL.COM resides only in directory MYDIR on drive B, and that the default drive is A. The commands

PATH B:MYDIR<CR> and

SPECIAL<CR>

would request TeleDOS to look for the command file SPECIAL.COM first on the current directory on drive A, then on MYDIR on drive B. If SPECIAL.COM had not been found in either directory, the following message is displayed:

Bad command or file name

If a path is specified that no longer exists, PATH ignores the path and goes on to the next path.

PRINT Command

Type External

Purpose Prints a queue of text files on a line printer while you are processing other TeleDOS commands (usually called "background printing").

Format PRINT [[<filespec>][/T][/C][/S]...]

Comments The PRINT command allows queuing up to ten text files for printing at one time. The * and ? global filename characters may be used in the filename and extension. Only files in the current directory can be queued for printing, but once the file has been queued, the directory can be changed without affecting the printing of queued files.

The first time the PRINT command is issued, the resident size of TeleDOS is increased by 3200 bytes.

/T TERMINATE: this parameter deletes all the files in the print queue (those waiting to be printed). If a file is currently being printed, the printing stops, a cancellation message is printed, the paper is advanced to the next page, and the printer's alarm sounds.

/C CANCEL: this parameter turns on the cancel mode. The preceding filespec and all following filespecs on the command line are canceled from the print queue until a /P parameter is found on the command line, or <CR> is pressed.

/P PRINT: this parameter turns on the print mode. The preceding filespec and all following filespecs on the command line are added to the print queue until a /C parameter is found on the command line, or <CR> is pressed.

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

Examples **PRINT /T<CR>**

empties the print queue.

PRINT /T *.ASM<CR>

empties the print queue and queues all .ASM files on the default drive.

PRINT Command

PRINT A:TEMP1.TST/C A:TEMP2.TST A:TEMP3.TST<CR>

removes the three files indicated from the print queue.

PRINT TEMP1.TST/C TEMP2.TST/P TEMP3.TST<CR>

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

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

Name of list device [PRN]:

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

List output is not assigned to a device

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

PRINT queue is full

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

PRINT queue is empty

There are no files in the print queue.

No files match d:XXXXXXXX.XXX

A filespec was given to add to the queue, but no file match the filespec.

NOTE! If there are no files in the queue to match the canceled filespec, no error message appears.

PRINT
Command

Drive not ready

If this message occurs when PRINT attempts a disk access, PRINT keeps trying until the drive is ready. Any other error causes the current file to be canceled. An error message would be output on your printer in such a case.

All files canceled

If the /T (TERMINATE) parameter is issued, the message "All files canceled by operator" is output on your printer. If the current file being printed is canceled by a /C, the message, <filename> canceled by operator, is printed.

PROMPT Command

Type Internal

Purpose Changes the TeleDOS command prompt.

Format PROMPT [<prompt>]

Comments This command allows you to change the TeleDOS system prompt (for example, A>). If no options are entered, the prompt is set to the default prompt, which is the default drive designation. You can set the prompt to a special prompt, such as the current time, by using the characters indicated below.

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

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

Examples **PROMPT ng<CR>** Sets the normal TeleDOS prompt (A>).

PROMPT Time = t_Date = $d<CR>

Sets a two-line prompt which prints:

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

RECOVER Command

Type External

Purpose Recovers a file or an entire disk containing bad sectors.

Format RECOVER <d>: or RECOVER <pathname>

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

To recover a particular file, enter:

RECOVER <pathname><CR>

TeleDOS reads the file sector by sector and skips the bad sector(s). When TeleDOS finds the bad sector(s), the sector(s) are marked and TeleDOS no longer allocates data to that sector.

To recover a disk, enter:

RECOVER <d>:<CR>

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

If there is not enough room in the root directory, RECOVER prints a message and stores information about the extra files in the File Allocation Table. You can run RECOVER again to regain these files when there is more room in the root directory.

REN (Rename) Command

Type Internal

Synonym RENAME

Purpose Changes the name of the first file listed to the second filename.

Format REN <pathname> <filename>

Comments The first file must be given a drive designation if the disk resides in a drive other than the default drive. Any drive designation for the second filename is ignored. The file remains on the disk where it currently resides.

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

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

REN *.LST *.TXT<CR>

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

REN B:ABODE ?D?B?<CR>

The file remains on drive B.

An attempt to rename a file to a name already present in the directory results in the following error message:

Duplicate file name or file not found

RMDIR (Remove Directory) Command

Type Internal

Synonym RD

Purpose Removes a sub-directory from a hierarchical directory structure.

Format RMDIR <path>

Comments This command removes a directory that is empty except for the . and .. shorthand symbols. The last directory name in the path is the directory removed.

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

RMDIR \BIN\USER\JOE<CR>

The directory is deleted from the directory structure.

SET Command

Type Internal

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

Format SET [<name>=[parameter]]

Comments The SET command inserts the entire string (name=parameter) into a block of memory reserved for environment strings. Any lower-case letters in the <name> are converted to upper-case letters. The <parameter> is inserted as it is entered. If <name> already exists in the environment, its <parameter> is replaced with the new <parameter>.

If the SET command is entered with no optional string, the current strings in the environment are displayed.

If the SET command is entered with <name>=, but no <parameter>, then the string with <name> is removed from the environment.

When you start up the operating system, TeleDOS places the string COMSPEC=\COMMAND.COM in the environment. This string describes the path TeleDOS uses to reload the command processor when necessary. If you use the PROMPT or PATH command, a string is added to the environment to indicate the prompt or path to use.

The SET command can also be used in batch processing. In this way, you can define your replaceable parameters with names instead of numbers. If your batch file contains the statement LINK %FILE%, you can set the name that TeleDOS uses for that variable with the SET command. The command:

SET FILE=DOMORE<CR>

replaces the %FILE% parameter with the filename DOMORE. Therefore, you do not need to edit each batch file to change the replaceable parameter names. Note that when you use text (instead of numbers) as replaceable parameters, the name must be ended by a percent sign.

SET Command

Example The command:

SET PROMPT=$n*

produces the same results as the command:

PROMPT $n*

Notes TeleDOS can expand the environment to hold additional strings if you have not loaded a program that remains resident (such as MODE, PRINT, or GRAPHICS). If you have loaded a program that remains resident, TeleDOS is unable to expand the environment area beyond 127 bytes. If the current environment is larger than 127 bytes when a program that remains resident is loaded, TeleDOS is unable to expand the environment beyond its current size.

SORT Command

Type External

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

Format SORT [/R] [/+n]

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

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

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

Examples The following command reads the file UNSORT.TXT, sorts the items in reverse order, and then writes the output to a file named SORT.TXT:

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

The following command pipes the output of the directory command to the SORT filter. The SORT filter sorts the directory listing starting with column 14 (this is the column in the directory listing that contains the file size), then sends the output to the screen. Thus, the result of this command is a directory sorted by file size:

DIR | SORT /+14<CR>

The command

DIR | SORT /+14 | MORE<CR>

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

SYS (System) Command

Type External

Purpose Transfers the TeleDOS system files from the disk in the default drive to the disk in the drive specified by d.

Format SYS <d>:

Comments The SYS command is normally used to place the operating system (or update the operating system) on a formatted disk which contains no files. An entry for d: is required.

If IO.SYS and MSDOS.SYS are on the destination disk, they must take up the same amount of space on the disk as the new system needs. This means that you cannot transfer system files from a TeleDOS disk (at 2.11) to an MS-DOS 1.1 disk. You must reformat the MS-DOS 1.1 disk with the TeleDOS FORMAT command before the SYS command will work.

The destination disk must be completely blank or already have the system files IO.SYS and MSDOS.SYS.

The transferred files are copied in the following order:

IO.SYS
MSDOS.SYS

IO.SYS and MSDOS.SYS are both hidden files that do not appear when the DIR command is executed. COMMAND.COM (the command processor) is not transferred. You must use the COPY command to transfer COMMAND.COM.

If SYS detects an error, one of the following messages will be displayed:

No room for system on destination disk

There is not enough room on the destination disk for the IO.SYS and MSDOS.SYS files.

Incompatible system size

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

TIME
Command

Type Internal

Purpose Displays and sets the time.

Format TIME [<hh>[:<mm>]]

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

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

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

TIME 8:20<CR>

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

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

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

TeleDOS uses the time entered as the new time if the options and separators are valid. If the options or separators are not valid, TeleDOS displays:

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

TeleDOS then waits for you to enter a valid time.

TREE Command

Type External

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

Format TREE [<d>:][/F]

Comments TREE displays the full path name of each directory found on the specified drive. If no drive option is included, the directories on the default drive are listed. If the /F parameter is included, TREE also lists the names of all the files found in each sub-directory.

Example Consider the following directory structure for a disk in drive B with the volume label DISK3.

```
                    ROOT DIR
                       |
                       |
          ----------------------------------
          |                                |
          |                                |
     SOURCE                            LEVEL2
          SOURCE1.ASM                    |  No files
          SOURCE2.ASM                    |
                                        LEVEL3
                                             BACKUP.SAV
```

The command:

TREE B:/F<CR>

produces the following display:

```
DIRECTORY PATH LISTING FOR VOLUME DISK3

Path:\SOURCE

Sub-directories: None

Files: SOURCE1.ASM
       SOURCE2.ASM
```

TREE Command

```
Path: \LEVEL2

Sub-directories: LEVEL3

Files: None

Path:\LEVEL2\LEVEL3

Sub-directories: None

Files: BACKUP.SAV
```

TYPE
Command

Type Internal

Purpose Displays the contents of the file on the screen.

Format TYPE <pathname>

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

Example **TYPE MYFILE.TXT | MORE<CR>**

displays the file MYFILE.TXT one screen at a time.

VER (Version) Command

Type Internal

Purpose Prints TeleDOS version number.

Format VER

Comments If you want to know what version of TeleDOS you are using, enter VER. The version number is displayed on your screen.

VERIFY Command

Type Internal

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

Format VERIFY [{ON|OFF}]

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

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

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

VOL (Volume) Command

Type Internal

Purpose Displays the disk volume label, if it exists.

Format VOL [<d>:]

Comments This command displays the volume label of the disk in the drive indicated. If no drive is specified, TeleDOS prints the volume label of the disk in the default drive.

If the disk does not have a volume label, VOL displays:

```
Volume in drive d has no label
```

BATCH PROCESSING COMMANDS

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

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

ECHO
Sub-Command

Type Internal

Purpose Turns batch echo feature on and off.

Format ECHO [{ON | OFF}] [<message>]

Comments Normally, commands in a batch file are displayed (echoed) on the screen as they are read from the batch file. ECHO OFF turns off this feature. ECHO ON turns the echo back on. ECHO ON is the power-on default setting.

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

ECHO <message> displays the message on the screen regardless of the current ON or OFF state.

BATCH PROCESSING COMMANDS

FOR
Sub-Command

Type Internal

Purpose Command extension used in batch and interactive file processing to allow repeated execution of TeleDOS commands.

Format FOR %%<c> IN (<set>) DO <command> (batch processing)

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

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

<set> is a list of items separated by spaces.

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

Example **FOR %%f IN (FILE1 FILE2 FILE3) DO DIR %%f**

is the same as:

DIR FILE1
DIR FILE2
DIR FILE3

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

BATCH PROCESSING COMMANDS

GOTO
Sub-Command

Type Internal

Purpose Command extension used in batch file processing.

Format GOTO <label>

Comments GOTO transfers control to the line after the <label> definition. If no label has been defined, the current batch file terminates and display the message:

```
Label not found
```

Example

```
:Loop
REM looping...
GOTO Loop
```

produces an infinite sequence of REM looping... and GOTO Loop messages on the screen.

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

BATCH PROCESSING COMMANDS

IF
Sub-Command

Type Internal

Purpose Command extension used in batch file processing.

Format IF [NOT] <condition> <command>

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

ERRORLEVEL <number>

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

<stringl> == <string2>

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

EXIST <filespec>

True if and only if <filespec> exists on the specified drive. A path is not allowed with the filespec.

NOT <condition>

True if and only if <condition> is false.

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

Examples IF NOT EXIST NEEDED.DAT ECHO Can't find file NEEDED.DAT

```
IF %1 == TEST GOTO CORRECT
     .
     .
:CORRECT
next command
```

BATCH PROCESSING COMMANDS

PAUSE
Sub-Command

Type Internal

Purpose Suspends execution of the batch file.

Format PAUSE [comment]

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

When the command processor encounters PAUSE, it prints:

```
PAUSE comment
Strike a key when ready . . .
```

If you press <Ctrl>/<Break> or <Ctrl>/<C>, another prompt will be displayed:

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

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

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

BATCH PROCESSING COMMANDS

REM (Remark)
Sub-Command

Type Internal

Purpose Displays the listed remark during execution of a batch file.

Format REM [comment]

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

Example

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

BATCH PROCESSING COMMANDS

SHIFT
Sub-Command

Type Internal

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

Format SHIFT

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

Example If a batch file MYPROG.BAT was started with the following command line:

MYPROG A B C D E F G H I J H

The replaceable parameters would be assigned as follows:

%0	MYPROG	%5	E
%1	A	%6	F
%2	B	%7	G
%3	C	%8	H
%4	D	%9	I

If a SHIFT command is executed in the batch file, the parameters would be shifted as follows:

%0	A	%5	F
%1	B	%6	G
%2	C	%7	H
%3	D	%8	I
%4	E	%9	J

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

6. TeleDOS EDITING AND FUNCTION KEYS

SPECIAL TeleDOS EDITING KEYS

TeleDOS provides special editing keys to aid in entering TeleDOS command lines and editing text lines while using the EDLIN line editor. The editing keys are used in conjunction with a template that contains the last command line, or text line entered. Figure 6-1 shows the relationship between an entered line of text and the template.

Figure 6-1
Text Line and Template

```
User Input
     |
     |
     |
     |
     V

Text Line <----------------> Template

     |
     |
     |
     V

EDLIN.COM
```

As shown in Figure 6-1, when a text line is entered and the <CR> key is pressed, the line is sent to EDLIN.COM. At the same time, a copy of this line is sent to the template. You can now recall the line or modify it with the TeleDOS special editing keys.

The following discussion refers to the use of the special editing keys while using the EDLIN line editor, but the features discussed equally apply to editing a TeleDOS command line.

Table 6-1 contains a complete list of the special editing keys.

Table 6-1
Special Editing Functions

Editing Key	Keyboard Key	Editing Function
<COPY1>	F1	Copies one character from the template
<COPYUP>	F2	Copies characters from the template up to but not including the character specified
<COPYALL>	F3	Copies all remaining characters from the template
<SKIPUP>	F4	Skips over, or moves the template pointer to the first occurrence of the specified character in the template
<NEWLINE>	F5	Makes the edited line the new template
<^Z>	F6	Puts a ^Z (1AH) end-of-file character in the new template
<INSERT>	Ins	Enters/exits insert mode
<SKIP1>	Del	Skips over, or moves the template pointer to the next character in the template
<VOID>	Esc	Voids the current input; leaves the template unchanged

The examples used in the following editing key descriptions would occur while editing a line in EDLIN. The editing functions equally apply while in the EDLIN I (insert) mode, or while editing a TeleDOS command line.

<COPY1> (F1)
Key

Purpose Copies one character from the template.

Comments Pressing the <COPY1> key copies one character from the template to the screen and turns off the insert mode.

Example Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <COPY1> key copies the first character (T) from the template to the screen.

```
         1:*This is a sample file
<COPY1> 1:*T_
```

Each time the <COPY1> key is pressed, one more character appears:

```
<COPY1> 1:*Th_
<COPY1> 1:*Thi_
<COPY1> 1:*This_
```

<COPYUP> (F2)
Key

Purpose To copy several characters from the template to the screen.

Comments Pressing the <COPYUP> key copies all characters up to but not including the first occurrence of a given character from the template to the screen. The given character is the next character typed after the <COPYUP> key is pressed; the character is not copied or displayed on the screen. Pressing the <COPYUP> key moves the template pointer to the given character in the template. If the template does not contain the specified character, nothing is copied. Pressing <COPYUP> also automatically turns off the insert mode.

Example Assume that the screen shows:

```
l:*This is a sample file.
l:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <COPYUP> key copies all characters up to the character specified immediately after the <COPYUP> key.

```
            l:*This is a sample file
<COPYUP>p l:*This is a sam_
```

<COPYALL> (F3)
Key

Purpose Copies the remainder of the template to the screen.

Comments Pressing the <COPYALL> key copies all the characters from the current template pointer position to the end of the template.

Example Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. The template pointer is pointing to the T in This. Pressing the <COPYALL> key copies all characters from the template (shown in the top line) to the line with the cursor (the lower line):

```
          1:*This is a sample file.  (template)
<COPYALL> 1:*This is a sample file._ (current line)
```

<COPYALL> automatically turns off the insert mode.

<SKIPUP> (F4)
Key

Purpose Skips multiple characters in the template up to the specified character.

Comments Pressing the <SKIPUP> key moves the template pointer to the first occurrence of the specified character. No characters are copied to the screen. If the template does not contain the specified character, the pointer is not moved. The action of the <SKIPUP> key is similar to the <COPYUP> key, except that <SKIPUP> skips over characters in the template rather than copying them to the screen.

Example Assume that the screen shows:

```
l:*This is a sample file.
l:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <SKIPUP> key skips over all the characters in the template up to the character pressed after the <SKIPUP> key:

```
            l:*This is a sample file
<SKIPUP>p l:*_
```

The cursor position does not change. To see how much of the line has been skipped over, press the <COPYALL> key to copy the remainder of the template to the screen.

```
          l:*This is a sample file:
<SKIPUP>p l:*_
<COPYALL> l:*ple file._
```

<NEWLINE> (F5)
Key

Purpose Creates a new template.

Comments Pressing the <NEWLINE> key copies the current line to the template. The contents of the old template are deleted. Pressing <NEWLINE> outputs an @ (at sign character), a carriage return, and a line feed. The current line is voided and the insert mode is turned off.

Example Assume that the screen shows:

```
l:*This is a sample file.
l:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Assume that you enter <COPYUP>m, <INSERT>lary, <INSERT> tax, and then <COPYALL>:

```
                l:*This is a sample file.
<COPYUP>m       l:*This is a sa_
<INSERT>lary    l:*This is a salary_
<INSERT> tax    l:*This is a salary tax_
<COPYALL>       l:*This is a salary tax file._
```

At this point, assume that you want this line to be the new template, so you press the <NEWLINE> key:

```
<NEWLINE>       l:*This is a salary tax file.@
                   _
```

The @ indicates that this new line is now the new template. Additional editing can be done using the new template.

<INSERT> (Ins) Key

Purpose Enters/exits insert mode.

Comments Pressing the <INSERT> key causes EDLIN to enter and exit the insert mode. The current screen cursor moves to the right as each character is entered, but the template pointer does not move.

Example Assume that the screen shows:

```
l:*This is a sample file.
l:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Assume that you press the <COPYUP> and f keys:

```
                l:*This is a sample file
<COPYUP>f l:*This is a sample _
```

Now press the <INSERT> key and insert the characters e, d, i, t, and one space:

```
                l:*This is a sample file.
<COPYUP>f       l:*This is a sample _
<INSERT>edit    l:*This is a sample edit _
```

If you now press the <COPYALL> key, the rest of the template is copied to the line:

```
                l:*This is a sample edit _
<COPYALL> l:*This is a sample edit file._
```

If you had pressed the <CR> key, the remainder of the template would have been truncated, and the current line would have ended at the end of the insert:

```
<INSERT>edit <CR> l:*This is a sample edit _
```

To exit the insert mode, press the <INSERT> key again.

<SKIP1> (Del)
Key

Purpose Skips over one character in the template.

Comments Pressing the <SKIP1> key moves the template pointer to the next character in the template without copying it to the screen.

Example Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <SKIP1> key skips over the first character (T).

```
        1:*This is a sample file
<SKIP1> 1:*_
```

The cursor position does not change and only the template is affected. To see how much of the line has been skipped over, press the <COPYALL> key, which copies the remainder of the template to the screen.

```
        1:*This is a sample file.
<SKIP1> 1:*_
<COPYALL> 1:*his is a sample file._
```

<VOID> (Esc)
Key

Purpose Ends input and voids the current line.

Comments Pressing the <VOID> key voids the current line, but leaves the template unchanged. <VOID> also prints a backslash (\), carriage return, and a line feed. The cursor (indicated by the underline) is positioned at the beginning of the line.

Example Assume that the screen shows:

```
l:*This is a sample file.
l:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Assume that you use the editing keys to change the line to "Sample File:"

```
l:*This is a sample file.
l:*Sample File_
```

Then you change your mind and decide to keep the original line. To cancel "Sample File" and keep "This is a sample file.", press <VOID>. Notice that a backslash appears on the Sample File line to tell you it has been cancelled.

```
        l:*This is a sample file.
<VOID>  l:*Sample File\
            _
```

Press <CR> or <COPYALL> and <CR> to keep the original line.

<VOID> automatically turns off the insert mode.

7. THE LINE EDITOR (EDLIN)

INTRODUCTION

In this chapter, you will learn how to use EDLIN, the line editor program. You can use EDLIN to create, change, and display files, whether they are source program or text files.

You can use EDLIN to:

Create new source files and save them

Update existing files and save both the updated and original files

Delete, edit, insert, and display lines

Search for, delete, or replace text within one or more lines

The text in files created or edited by EDLIN is divided into lines, each up to 254 characters long. Line numbers are generated and displayed by EDLIN during the editing process, but are not actually present in the saved file.

When you insert lines, all line numbers following the inserted text advance automatically by the number of lines being inserted. When you delete lines in a file, all line numbers following the deleted text decrease automatically by the number of lines deleted. As a result, lines are always numbered consecutively in your file.

HOW TO START EDLIN

To start EDLIN, enter:

EDLIN [<path>]<filename>[.<ext][/B]<CR>

If you are creating a new file, the <filename> should be the name of the file you wish to create. If EDLIN does not find this file, EDLIN creates a new file with the name you specify. The following message and prompt are displayed:

```
New file
*_
```

Notice that the prompt for EDLIN is an asterisk (*).

You can now enter lines of text into your new file. To begin entering text, you must enter an I (Insert) command to insert lines. The I command is discussed later in this chapter.

If you want to edit an existing file, <filespec> should be the name of the file you want to edit. When EDLIN finds the file you specify on the designated or default drive, the file is loaded into memory. If the entire file can be loaded, EDLIN displays the following message on your screen:

```
End of input file
*
```

You can then edit the file using EDLIN editing commands.

If the file is too large to be loaded into memory, EDLIN loads lines until memory is 3/4 full, then displays the * prompt. You can then edit the portion of the file that is in memory.

To edit the remainder of the file, you must save some of the edited lines on disk to free memory; then EDLIN can load the unedited lines from disk into memory. Refer to the Write and Append commands in this chapter for the procedure.

When you complete the editing session, you can save the original and the updated (new) files by using the End command. The End command is discussed later in this chapter. The original file is renamed with a .BAK extension, and the new file has the filename and extension you specify in the EDLIN command. The original .BAK file is not erased until the end of the editing session, or until disk space is needed by the editor (EDLIN).

Do not try to edit a file with a filename extension of .BAK. EDLIN assumes that any .BAK file is a backup file. If you find it necessary to edit such a file, rename the file with another extension (using the TeleDOS RENAME command discussed in Chapter 5), then start EDLIN and specify the new <filespec>.

The special editing keys and template discussed in Chapter 6 are used to edit your text files.

COMMAND INFORMATION

EDLIN commands perform editing functions on lines of text. The following list contains information you should read before you use EDLIN commands.

1. Pathnames are acceptable as options to commands. For example, entering EDLIN \BIN\USER\JOE\TEXT.TXT allows you to edit the TEXT.TXT file in the subdirectory JOE.

2. You can reference line numbers relative to the current line (the line with the asterisk). Use a minus sign with a number to indicate lines before the current line. Use a plus sign with a number to indicate lines after the current line.

   Example:

   -10,+10L

   This command lists 10 lines before the current line, the current line, and 10 lines after the current line.

3. Multiple commands may be issued on one command line. When you issue a command to edit a single line using a line number (<line>), a semicolon must separate commands on the line. Otherwise, one command may follow another without any special separators. In the case of a Search or Replace command, the <string> may be ended by a <^Z> instead of a <CR>.

   Examples:

   The following command line edits line 15 and then displays lines 10 through 20 on the screen.

   15;-5,+5L

   The command line in the next example searches for "This string" and then displays five lines before and five lines after the line containing the matched string. If the search fails, then the displayed lines are those line numbers relative to the current line.

   SThis string^Z-5,+5L

4. You can enter EDLIN commands with or without a space between the line number and command. For example, to delete line 6, the command 6D is the same as 6 D.

5. It is possible to insert a control character (such as ^Z) into text by using the quote character ^V before it while in the insert mode. ^V tells TeleDOS to recognize the next capital letter entered as a control character. It is also possible to use a control character in any of the string arguments of Search or Replace by using the special quote character. For example:

S^VZ	looks for the first occurrence of ^Z in a file
R^VZ^Ztest	replaces all occurrences of ^Z in a file by test

S^VC^Zbar replaces all occurrences of ^C by bar

It is possible to insert ^V into the text by entering ^VV.

6. The ^Z character ordinarily tells EDLIN that this is the end of the file. If you have ^Z characters elsewhere in your file, you must tell EDLIN that these other control characters do not mean end-of-file. Use the /B parameter to tell EDLIN to ignore any ^Z characters in the file and to show you the entire file.

The EDLIN commands are summarized in the following table. They are also described in more detail following the description of command options.

Table 7-1
EDLIN Commands

Command	Purpose
<line>	Edits line number <line>
A	Appends lines
C	Copies lines
D	Deletes lines
E	Ends editing
I	Inserts lines
L	Lists lines
M	Moves lines
P	Pages text
Q	Quits editing
R	Replaces lines
S	Searches text
T	Transfers text
W	Writes lines

Command Options

Several EDLIN commands accept one or more options. The effect of a command option varies, depending on which command it is used with. The following list describes each option.

<line> <line> indicates a line number that you enter. Line numbers must be separated from other line numbers by a comma or a space.

<line> may be specified one of three ways:

Number Any number less than 65534. If a number larger than the largest existing line number is specified, then <line> means the line after the last line number.

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

Pound(#) The pound sign indicates the line after the last line number. If you specify # for <line>, this has the same effect as specifying a number larger than the last line number.

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

? The question mark option directs EDLIN to ask you if the correct string has been found. The question mark is used only with the Replace and Search commands. Before continuing, EDLIN waits for either a Y or <CR> for a yes response, or for any other key for a no response.

<string> <string> represents text to be found, to be replaced, or to replace other text. The <string> option is used only with the Search and Replace commands. Each <string> must be ended by a <^Z> or a <CR> (see the Replace command for details). No spaces should be left between strings or between a string and its command letter, unless you want those spaces to be part of the string.

EDLIN COMMANDS

The following pages describe EDLIN editing commands.

A (Append) Command

Purpose Adds the specified number of lines from disk to the file being edited in memory. The lines are added at the end of lines that are currently in memory.

Format [<n>]A

Comments This command is meaningful only if the file being edited is too large to fit into memory. As many lines as possible are read into memory for editing when you start EDLIN.

To edit the remainder of the file that did not fit into memory, lines that have already been edited must be written to disk. Then you can load unedited lines from disk into memory with the Append command. Refer to the Write command in this chapter for information on how to write edited lines to disk.

Notes

1. If you do not specify the number of lines to append, lines are appended to memory until available memory is 3/4 full. No action is taken if available memory is already 3/4 full.

2. The message "End of input file" is displayed when the Append command has read the last line of the file into memory.

C (Copy) Command

Purpose Copies a range of lines to a specified line number. The lines can be copied as many times as you want by using the <count> option.

Format [<line>],[<line>],<line>[,<count>]C

Comments If you do not specify a number in <count>, EDLIN copies the lines one time. If the first and/or the second <line> is omitted, the default is the current line. The file is renumbered automatically after the copy.

The line numbers must not overlap or the error message, Entry error, is displayed. For example, 3,20,15C would result in an error message.

To copy a single line, the line number must be listed twice; once as the first line number in the group, and once as the last line number in the group.

NOTE! The first and second <line> entries are optional, but the commas are not.

Examples Assume that the following file exists and is ready to edit:

```
1: This is a sample file
2: used to show copying lines.
3: See what happens when you use
4: the Copy command
5: (the C command)
6: to copy text in your file.
```

You can copy this entire block of text by issuing the following command:

1,6,7C<CR>

C (Copy) Command

The result is:

```
 1: This is a sample file
 2: used to show copying lines.
 3: See what happens when you use
 4: the Copy command
 5: (the C command)
 6: to copy text in your file.
 7:*This is a sample file
 8: used to show copying lines.
 9: See what happens when you use
10: the Copy command
11: (the C command)
12: to copy text in your file.
```

If you want to place text within other text, the third <line> entered should specify the new line number of the first line of copied text. For example, assume that you want to copy lines and insert them within the following file:

```
 1: This is a sample file
 2: used to show copying lines.
 3: See what happens when you use
 4: the Copy command
 5: (the C command)
 6: to copy text in your file.
 7: You can also use COPY
 8: to copy lines of text
 9: to the middle of your file.
10: End of sample file.
```

The command 3,6,10C results in the following file:

```
 1: This is a sample file
 2: used to show copying lines.
 3: See what happens when you use
 4: the Copy command
 5: (the C command)
 6: to copy text in your file.
 7: You can also use COPY
 8: to copy lines of text
 9: to the middle of your file.
10:*See what happens when you use
11: the Copy command
12: (the C command)
13: to copy text in your file.
14: End of sample file.
```

D (Delete) Command

Purpose Deletes a specified range of lines in a file.

Format [<line>][,<line>]D

Comments If the first <line> is omitted, that option defaults to the current line (the line with the asterisk next to the line number). If the second <line> is omitted, then just the first <line> is deleted. When lines have been deleted, the line immediately after the deleted section becomes the current line and has the same line number as the first deleted <line> had before the deletion occurred.

Examples Assume that the following file exists and is ready to edit:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 .
 .
 .
25: (the D and I commands)
26: to edit the text
27:*in your file.
```

To delete multiple lines, enter:

5,24D<CR>

The result is:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5:*(the D and I commands)
 6: to edit the text
 7: in your file.
```

To delete a single line, enter:

6D<CR>

D (Delete) Command

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6:*in your file.
```

Next, to delete a range of lines from the following file starting at the current line:

```
1: This is a sample file
2: used to show dynamic line numbers.
3:*See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.
```

Enter:

,6D<CR>

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3:*in your file.
```

Notice that the lines are automatically renumbered.

Edit Command

Purpose Edits line of text.

Format [<line>]

Comments When a line number is entered, EDLIN displays the line number and text; then, on the line below, EDLIN reprints the line number. The line is now ready for editing. You may use any of the EDLIN editing commands to edit the line. The existing text of the line serves as the template until <CR> is pressed.

If no line number is entered, the line after the line marked with an asterisk (*) is edited. If no changes to the current line are needed and the cursor is at the beginning or end of the line, press <CR> to accept the line as is.

NOTE! If <CR> is pressed while the cursor is in the middle of the line, the remainder of the line is deleted.

If you are editing a line and decide to void the edited version and retain the original line, press <Ctrl>/<Break> or <Esc>.

Example Assume that the following file exists and is ready to edit:

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

To edit line 4, enter:

4<CR>

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

```
4:*four.
4:*_
```

Now, using the <INSERT> and <COPYALL> special editing keys, enter:

```
<INSERT>number      4:*number _
<COPYALL><CR>       4:*number four.
                  *
```

E (End) Command

Purpose Ends the editing session.

Format E

Comments This command saves the edited file on disk, renames the original input file <filename>.BAK, and then exits EDLIN. If the file was created during the editing session, no .BAK file is created.

The E command takes no options. Therefore, you cannot tell EDLIN on which drive to save the file. The drive you want to save the file on must be selected when the editing session is started. If the drive is not selected when EDLIN is started, the file is saved on the disk in the default drive. The file can be copied to a different drive using the TeleDOS COPY command.

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

Example **E<CR>**

After execution of the E command, the TeleDOS default drive prompt (for example, A>) is displayed.

I (Insert) Command

Purpose Inserts text immediately before the specified <line>.

Format [<line>]I

Comments If you are creating a new file, the I command must be given before text can be entered (inserted). Text begins with line number 1. Successive line numbers appear automatically each time <CR> is pressed.

EDLIN remains in insert mode until <Ctrl>/<Break> or <^C> is entered. When the insert is completed and the insert mode has been exited, the line immediately following the inserted lines becomes the current line. All line numbers following the inserted section are incremented by the number of lines inserted.

If <line> is not specified, the default is the current line number and lines are inserted immediately before the current line. If <line> is any number larger than the last line number, or if a pound sign (#) is specified as <line>, the inserted lines are appended to the end of the file. After appending lines, a current line does not exist.

Examples Assume that the following file exists and is ready to edit:

```
1:*This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.
```

To insert text before a specific line that is not the current line, enter <line>I:

7I<CR>

The result is:

7:*_

Now, enter the text for line 7:

7:***and renumber lines<CR>**

Then to end the insertion, press <^C> on the next line:

8:*<^C>

I (Insert) Command

Now enter L to list the file. The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: and renumber lines
8:*in your file.
```

To insert lines immediately before the current line, enter:

I<CR>

The result is:

```
8:*_
```

Now, insert the following text and terminate with a <^C> on the next line:

```
8:*so they are consecutive<CR>
9:*<^C>
```

Now to list the file and see the result, enter:

L<CR>

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: and renumber lines
8: so they are consecutive
9:*in your file.
```

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

10I<CR>

This produces the following:

```
10:*_
```

I (Insert) Command

Now, enter the following new lines:

```
10:*The insert command can place new lines<CR>
11:*in the file; there's no problem<CR>
12:*because the line numbers are dynamic;<CR>
13:*they'll go all the way to 65533.<CR>
14:*<^C>
```

End the insertion by pressing <^C> on line 14. The new lines appear at the end of the file. Now enter the List command:

```
L<CR>
```

The result is:

```
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: and renumber lines
 8: so they are consecutive
 9: in your file.
10: The insert command can place new lines
11: in the file; there's no problem
12: because the line numbers are dynamic;
13: they'll go all the way to 65533.
```

L (List) Command

Purpose Lists a range of lines, including the two lines specified.

Format [<line>][,<line>]L

Comments Default values are provided if either one or both of the options are omitted. If you omit the first option, as in:

,<line>L<CR>

the display starts eleven lines before the current line and ends with the specified <line>. The beginning comma is required to indicate the omitted first option.

NOTE! If the specified <line> is more than eleven lines after the current line, the display is the same as if you omitted both options.

If you omit the second option, as in

<line>L<CR>

23 lines are displayed, starting with the specified <line>.

If you omit both parameters, as in

L<CR>

23 lines will be displayed, the eleven lines before the current line, the current line, and the eleven lines after the current line. If there are less than eleven lines before the current line, more than 11 lines after the current line are displayed to make a total of 23 lines.

L (List)
Command

Examples Assume that the following file exists and is ready to edit:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
  .
  .
  .
15:*The current line contains an asterisk.
  .
  .
  .
26: to edit text
27: in your file.
```

To list a range of lines without reference to the current line, enter <line>,<line>L:

2,5L<CR>

The result is:

```
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
```

To list a range of 23 lines centered around the current line, enter only L:

L<CR>

The result is:

```
 4: Delete and Insert
 5: (the D and I commands)
  .
  .
  .
13: The current line is listed in the middle.
14: The current line remains unchanged.
15:*The current line contains an asterisk.
  .
  .
  .
26: to edit text.
```

M (Move) Command

Purpose Moves a range of text to the line specified.

Format [<line>],[<line>],<line>M

Comments Use the Move command to move a block of text (from the first <line> to the second <line>) to the location indicated by the third <line>. The lines are renumbered according to the direction of the move.

Examples To move lines 20-30 to line 100, enter:

20,30,100M<CR>

To move 25 lines starting at the current line, enter:

,+24,100M<CR>

If the line numbers overlap, EDLIN displays an Entry error message.

P (Page) Command

Purpose Lists the specified block of lines.

Format [<line>][,<line>]P

Comments This command pages through the block of specified lines. If first <line> is omitted, that number defaults to the current line plus one. If the second <line> is omitted, 23 lines are listed. If both <line>s are omitted, 23 lines are listed starting at the line after the current line.

Page differs from the List command in that it changes the current line. The last line displayed by the page command becomes the new current line.

Examples The command:

10,56P<CR>

scrolls through lines 10 to 56. To suspend the scrolling, press <Ctrl>/<Num Lock>. To continue the scrolling, press any key.

The command:

25P<CR>

lists 23 lines, starting at line 25.

Q (Quit) Command

Purpose Ends the editing session and exits to the TeleDOS operating system without saving the edited version of the file.

Format Q

Comments EDLIN prompts you to make sure you don't want to save the changes.

Enter Y if you want to end (Quit) the editing session. No editing changes are saved and no .BAK file is created. Refer to the End command in this chapter for information about the .BAK file.

Enter N or any other character except Y if you want to continue the editing session.

NOTE! When started, EDLIN erases any previous copy of the file with an extension of .BAK to make room to save the new copy. If you reply Y to the Abort edit (Y/N)? message, your previous backup copy will no longer exist.

Example

```
*Q
Abort edit (Y/N)? Y
A>_
```

A <CR> is not needed after entering Y or N in response to the abort edit prompt.

R (Replace) Command

Purpose Replaces all occurrences of a string of text in the specified range with a different string of text or blanks.

Format [<line>][,<line>][?]R[<string1>][^Z<string2>]

Comments As each occurrence of <string1> is found, it is replaced by <string2>. Each line with a replacement 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 of lines are replaced by <string2>, the R command terminates and the asterisk prompt reappears.

If the first <line> is omitted in the range argument (as in ,<line>) then the first <line> defaults to the line after the current line. If the second <line> is omitted (as in <line> or <line>,), the second <line> defaults to #. Therefore, this is the same as <line>,#. Remember that # indicates the line after the last line of the file.

If <string1> is omitted, Replace takes the search string from the most recent S or R command as its value. If there is no old <string1> (this is the first replace done), then the replacement process terminates immediately.

If a second string is to be given as a replacement, <string1> must be separated from <string2> with a ^Z.

If <string1> is ended with a ^Z and there is no <string2>, <string2> is taken as an empty string and a delete is performed. For example,

R<string1>^Z<CR>

deletes occurrences of <string1>, but

R<string1><CR> and
R<CR>

replaces <string1> by the old <string2> and the old <string1> with the old <string2>, respectively. Note that old refers to a previous string specified either in a Search or a Replace command.

R (Replace) Command

If the question mark (?) option is used, the Replace command stops at each line with a string that matches <string1>, displays the line with <string2> in place, and then displays the prompt O.K.?. If you press Y or a <CR>, <string2> replaces <string1> and the next occurrence of <string1> is searched for. This process continues until the end of the range or until the end of the file. After the last occurrence of <string1> is found, EDLIN displays the asterisk prompt.

If you press any key besides Y or <CR> after the O.K.? prompt, <string1> is left as it was in the line, and Replace goes to the next occurrence of <string1>. If <string1> occurs more than once in a line, each occurrence of <string1> is replaced individually, and the O.K.? prompt is displayed after each replacement. In this way, only the desired <string1> is replaced, and you can prevent unwanted substitutions.

Examples Assume that the following file exists and is ready for editing:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: in your file.
 8: The insert command can place new lines
 9: in the file; there's no problem
10: because the line numbers are dynamic;
11: they'll go all the way to 65533.
```

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

2,12Rand^Zor<CR>

The result is:

```
 4: Delete or Insert
 5: (the D or I commors)
 8: The insert commor can place new lines
```

Note that in the above replacement, some unwanted substitutions have occurred. To avoid these and to confirm each replacement, the same original file can be used with a slightly different command.

R (Replace) Command

In the next example, to replace only certain occurrences of the first <string> with the second <string>, enter:

```
2?Rand^Zor<CR>
```

The result is:

```
4: Delete or Insert
O.K.? Y
5: (The D or I commands)
O.K.? Y
5: (The D or I commors)
O.K.? N
8: The insert commor can place new lines
O.K.? N
*_
```

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

```
    .
    .
4: Delete or Insert
5: (The D or I commands)
    .
8: The insert command can place new lines
    .
    .
```

S (Search) Command

Purpose Searches the specified range of lines for a specified string of text.

Format [<line>][,<line>][?]S<string><CR>

Comments Search looks for an exact match of upper-case and lower-case characters. The first line that matches <string> is displayed and becomes the current line. If the question mark option is not specified, the Search command terminates when a match is found. If no line contains a match for <string>, the message, Not found, is displayed and the current line is not changed.

If the question mark option (?) is included in the command, EDLIN displays the first line with a matching string; it then prompts you with the message O.K.?. If you press either Y or <CR>, 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 (and the Not found message is displayed).

If the first <line> option is omitted, 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 # (the line after the last line in the file), which is the same as <line>,#S<string>. If <string> is omitted, Search takes the old string if there is one. (Note that old refers to the last search string specified in a previous Search or Replace command.) If there is not an old string (no previous search or replace has been done), the command terminates immediately.

Examples Assume that the following file exists and is ready for editing:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: in your file.
 8: The insert command can place new lines
 9: in the file; there's no problem
10: because the line numbers are dynamic;
11:*they'll go all the way to 65533.
```

S (Search) Command

To search for the first occurrence of the string "and", enter:

1Sand<CR>

The following line is displayed:

4: Delete and Insert

To get the "and" in line 5, modify the search command by entering:

,12 Sand<CR>

The search then continues from the line after the current line (line 4), since a first line was not given. The result is:

5: (the D and I commands)

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

1,?Sand<CR>

The result is:

```
4: Delete and Insert
O.K.?_
```

Press N (or any key except Y or <CR>) to continue the search.

O.K.? **N**

Continue:

```
5: (the D and I commands)
O.K.?_
```

Now press Y to terminate the search:

O.K.? **Y**
*_

S (Search) Command

To search for string XYZ without the verification (O.K.?), enter:

SXYZ<CR>

EDLIN reports the first match. To continue to search for the same string, enter:

S<CR>

EDLIN searches for the next match.

Note that <string> defaults to the string specified by the previous Replace or Search command.

T (Transfer) Command

Purpose Inserts (merges) the contents of <filename> into the file currently being edited at <line>. If <line> is omitted, then the current line is used.

Format [<line>]T<filespec>

Comments This command is useful if you want to put the contents of a file into another file or into the text you are editing. The transferred text is inserted at the line number specified by <line> and the lines are renumbered.

Example The command:

120TB:MYFILE.TXT<CR>

merges the file MYFILE.TXT from drive B into the current file at line 120.

W (Write) Command

Purpose Writes a specified number of lines to disk from the lines that are being edited in memory. Lines are written to disk beginning with line number 1.

Format [<n>]W

Comments This command is meaningful only if the file you are editing is too large to fit into memory. When you start EDLIN, EDLIN reads lines into memory until memory is 3/4 full.

To edit the remainder of your file, you must write edited lines in memory to disk. Then you can load additional unedited lines from disk into memory by using the Append command.

NOTE! If you do not specify the number of lines, lines are written to disk until 25% of memory is again available for adding text. No action is taken if 25% of memory is currently available. All lines are renumbered, so that the first remaining line becomes line number 1.

ERROR MESSAGES

When EDLIN finds an error, one of the following error messages is displayed:

Cannot edit .BAK file--rename file

Cause: You attempted to edit a file with a filename extension of .BAK. .BAK files cannot be edited because this extension is reserved for backup copies.

Cure: If you need the .BAK file for editing purposes, you must either RENAME the file with a different extension; or COPY the .BAK file and give it a different filename extension.

No room in directory for file

Cause: When you attempted to create a new file, either the file directory was full or you specified an illegal disk drive or an illegal filename.

Cure: Check the command line that started EDLIN for illegal filename or disk drive entries. If the command is no longer on the screen and if you have not yet entered a new command, the EDLIN start command can be recovered by pressing the <COPYALL> key.

If this command line contains no illegal entries, run the CHKDSK program on the specified disk drive. If the status report shows that the disk directory is full, remove the disk. Insert and format a new disk.

Entry Error

Cause: The last command entered contained a syntax error.

Cure: Reenter the command with the correct syntax (format) and press <CR>.

Line too long

Cause: During a Replace command, the string given as the replacement caused the line to expand beyond the limit of 253 characters. EDLIN aborted the Replace command.

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

ERROR MESSAGES

Disk Full--file write not completed

Cause: You gave the End command, but the disk did not contain enough free space for the whole file. EDLIN aborted the E command and returned you to the operating system. Some of the file may have been written to the disk.

Cure: Only a portion (if any) of the file has been saved. You should probably delete that portion of the file and restart the editing session. The file will not be available after this error. Always be sure that the disk has sufficient free space for the file to be written to disk before you begin your editing session.

Incorrect DOS version

Cause: You attempted to run EDLIN under a version of TeleDOS that was not 2.0 or higher.

Cure: You must make sure that the version of TeleDOS that you are using is 2.0 or higher.

Invalid drive name or file

Cause: You have not specified a valid drive or filename when starting EDLIN.

Cure: Specify the correct drive or filename.

Filename must be specified

Cause: You did not specify a filename when you started EDLIN.

Cure: Specify a filename in the EDLIN command line.

Invalid Parameter

Cause: You specified a parameter other than /B when starting EDLIN.

Cure: /B is the only valid EDLIN parameter.

Insufficient memory

Cause: There is not enough memory to run EDLIN.

Cure: You must free some memory by writing files to disk or by deleting files before restarting EDLIN.

ERROR MESSAGES

File not found

Cause: The filename specified during a Transfer command was not found.

Cure: Specify a valid filename when issuing a Transfer command.

Must specify destination number

Cause: A destination line number was not specified for a Copy or Move command.

Cure: Reissue the command with a destination line number.

Not enough room to merge the entire file

Cause: There was not enough room in memory to hold the file during a Transfer command.

Cure: You must free some memory by writing some lines to disk or by deleting some lines before you can transfer this file.

File creation error

Cause: The EDLIN temporary file cannot be created.

Cure: Check to make sure that the directory has enough space to create the temporary file. Also, make sure that the file does not have the same name as a subdirectory in the directory where the file to be edited is located.

8. FILE COMPARISON UTILITY (COMP)

INTRODUCTION

It is sometimes useful to compare files on your disk. If you have copied a file and later want to compare copies to see which one is current, you can use the TeleDOS File Comparison Utility (COMP).

COMP compares the contents of two files. The differences between the two files can be output to the screen or to a third file. The files being compared may be either source files (files containing source statements of a programming language); or binary files.

The comparisons are made in one of two ways: on a line-by-line or a byte-by-byte basis. The line-by-line comparison isolates blocks of lines that are different between the two files and displays those blocks of lines. The byte-by-byte comparison displays the bytes that are different between the two files.

Limitations On Source Comparisons

COMP uses a large amount of memory as buffer (storage) space to hold the source files. If the source files are larger than available memory, COMP compares what can be loaded into the buffer space. If no lines match in the portions of the files in the buffer space, COMP displays only the message:

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

For binary files larger than available memory, COMP compares both files completely, overlaying the portion in memory with the next portion from disk. All differences are output in the same manner as those files that fit completely in memory.

FILE SPECIFICATIONS

All file specifications use the following format:

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

where: d: is the letter designating a disk drive. If the drive designation is omitted, COMP defaults to the operating system's (current) default drive.

filename is a one- to eight-character name of the file.

.ext is a one- to three-character extension to the filename.

HOW TO USE COMP

The format of the COMP command is as follows:

COMP [/#][/B][/W][/C] <filenamel> <filename2>

COMP matches the first file <filenamel> against the second <filename2> and reports any differences between them. Both filenames can be pathnames. For example,

COMP B:\FOO\BAR\FILEl.TXT \BAR\FILE2.TXT

COMP takes FILEl.TXT in the \FOO\BAR directory of disk drive B and compares it with FILE2.TXT in the \BAR directory. Since no drive is specified for filename2, COMP assumes that the \BAR directory is on the disk in the default drive.

COMP PARAMETERS

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

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

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

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

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

/# # stands for a number from 1 to 9. This parameter specifies the number of lines required to match for the files to be considered as matching again after a difference has been found. If this parameter is not specified, it defaults to 3. This parameter is used only in source comparisons.

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

```
___More__data_to_be_found____
```

matches with

```
More_data_to_be_found
```

and with

```
_______More_______data_to_be_____found_______
```

but does not match with

```
____Moredata_to_be_found
```

This parameter is used only in source comparisons.

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

```
Much_MORE_data_IS_NOT_FOUND
```

matches

```
much_more_data_is_not_found
```

If both the /W and /C options are specified, then COMP compresses whites and ignore case. For example,

```
___DATA_was_found____
```

matches

```
data_was_found
```

This parameter is used only in source comparisons.

DIFFERENCE REPORTING

The File Comparison Utility reports the differences between the two files you specify by displaying the first filename, followed by the lines that differ between the files, followed by the first line to match in both files. COMP then displays the name of the second file followed by the lines that are different, followed by the first line that matches. The default for the number of lines to match between the files is 3. (If you want to change this default, specify the number of lines with the /# parameter.) For example:

```
          ...
          ...

----------<filename1>
<difference>
<1st line to match file2 in file1>

----------<filename2>
<difference>
<1st line to match file1 in file2>

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

          ...
          ...
```

COMP continues to list each difference.

If there are too many differences (involving too many lines), the program reports that the files are different and stops.

If no matches are found after the first difference is found, COMP displays:

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

and returns to the TeleDOS default drive prompt (for example, A>).

If the two files are the same, no messages are displayed and the TeleDOS default prompt is returned (A>).

REDIRECTING COMP OUTPUT TO A FILE

The differences and matches between the two files you specify are displayed on your screen unless you redirect the output to a file. This is accomplished in the same way as TeleDOS command redirection (refer to Chapter 4, Learning About Commands).

To compare File1 and File2 and then send the COMP output to DIFFER.TXT, enter:

COMP File1 File2 >DIFFER.TXT

The differences and matches between File1 and File2 are placed in file DIFFER.TXT on the default drive.

EXAMPLES

Example 1:

Assume these two ASCII files are on disk:

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

To compare the two files and display the differences on the terminal screen, enter:

COMP ALPHA.ASM BETA.ASM<CR>

COMP compares ALPHA.ASM with BETA.ASM and displays the differences on the terminal screen. All other defaults remain intact. (The defaults are: do not use tabs, spaces, or comments for matches, and do a source comparison on the two files.)

The output on the terminal screen appears as follows (the Notes do not appear):

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

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

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

----------ALPHA.ASM
M                          NOTE: ALPHA file
N                          contains mno where
O                          BETA contains jl2.
P

----------BETA.ASM
J
l
2
P

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

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

Example 2:

You can print the differences on the line printer using the same two source files. In this example, four successive lines must be the same to constitute a match.

Enter:

COMP /4 ALPHA.ASM BETA.ASM >PRN<CR>

The following output is sent to the printer:

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

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

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

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

Example 3:

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

Enter:

COMP /B ALPHA.ASM BETA.ASM<CR>

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

```
--ADDRS----F1---F2--
 00000009   44   47
 0000000C   45   48
 0000000F   46   49
 00000012   47   4A
 00000015   48   31
 00000018   49   32
 0000001B   4D   50
 0000001E   4E   51
 00000021   4F   52
 00000024   50   53
 00000027   51   54
 0000002A   52   55
 0000002D   53   56
 00000030   54   34
 00000033   55   35
 00000036   56   57
 00000039   57   58
 0000003C   58   59
 0000003F   59   5A
 00000042   5A   1A

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

ERROR MESSAGES

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

Incorrect DOS version

You are running COMP under a version of DOS that is not 2.0 or higher.

Invalid parameter:<option>

One of the parameters that you have specified is invalid.

File not found:<filename>

COMP could not find the filename you specified.

Read error in:<filename>

COMP could not read the entire file.

Invalid number of parameters

You have specified the wrong number of options on the COMP command line.

9. THE LINKER PROGRAM

INTRODUCTION

This chapter describes the TeleDOS linker program. It is recommended that you read the entire chapter before using the linker program.

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

The linker is a program that:

Combines separately produced object modules into one relocatable load module, a program you can run

Searches library files for definitions of unresolved external references

Resolves external cross-references

Produces a listing that shows both the resolution of external references and error messages

OVERVIEW OF THE LINKER PROGRAM

When you write a program, you write it in source code. This source code is passed through a compiler which produces object modules. The object modules must be passed through the link process to produce machine language that the computer can understand directly. This machine language is in the form required for running programs.

You may wish to link (combine) several programs and run them together. Each of your programs may refer to a symbol that is defined in another object module. This reference is called an external reference.

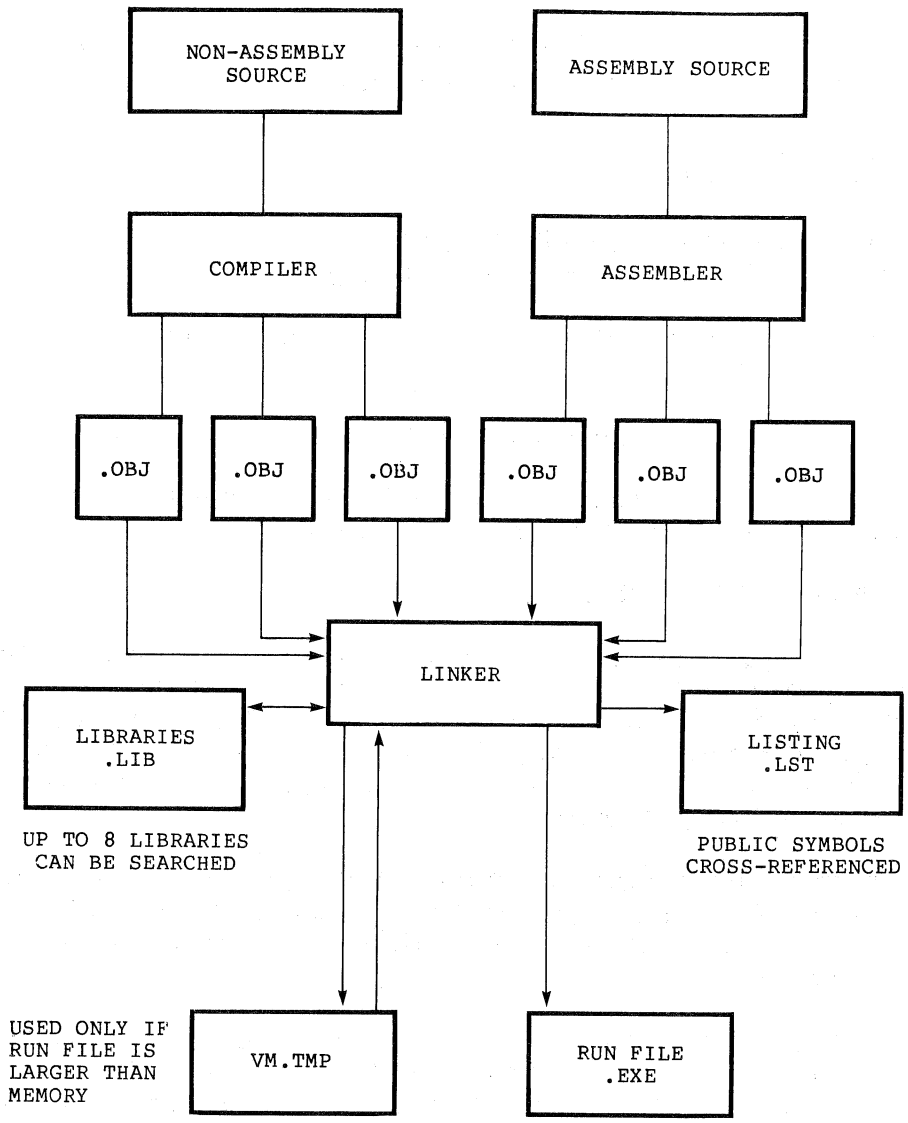
The linker program combines several object modules into one relocatable load module, or Run file (called an .EXE or Executable file). As it combines modules, the linker makes sure that all external references between object modules are defined. The linker can search several library files for definitions of any external references that are not defined in the object modules.

The linker also produces a List file that shows external references resolved, and it also displays any error messages.

The linker uses memory until available memory is exhausted, then creates a temporary disk file named VM.TMP.

Figure 9-1 illustrates the various parts of the linking operation.

Figure 9-1
Linking Operation

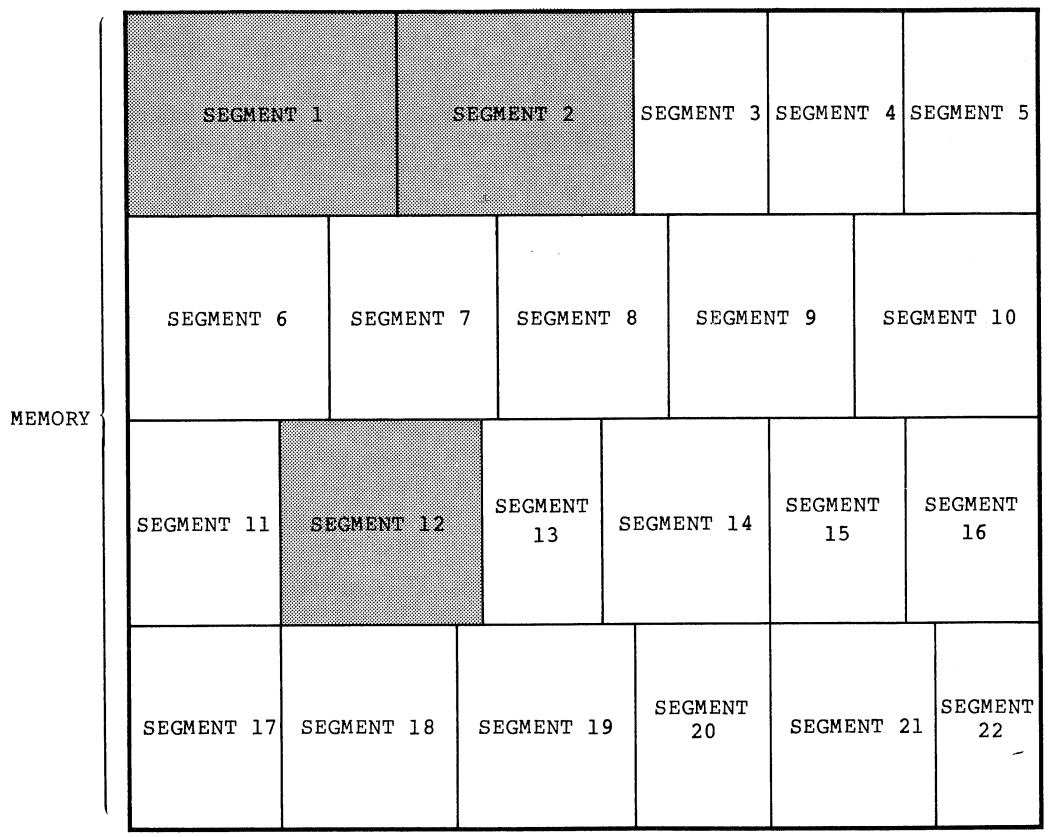


DEFINITIONS YOU NEED TO KNOW

Some of the terms used in this chapter are explained to help you understand how the linker works. Generally, if you are linking object modules compiled from BASIC, Pascal, or a high-level language, you do not need to know these terms. If however, you are writing and assembling programs in assembly language, you need to understand the definitions described below.

In TeleDOS, memory can be divided into segments, classes, and groups. Figure 9-2 illustrates these concepts.

Figure 9-2
How Memory Is Divided



shaded area = a group (64K bytes addressable)

Example:

	Segment Name	Segment Class Name
Segment 1	PROG.1	CODE
Segment 2	PROG.2	CODE
Segment 12	PROG.3	DATA

Note that segments 1, 2, and 12 have different segment names but may or may not have the same segment class name. Segments 1, 2, and 12 form a group with a group address of the lowest address of segment 1.

Each segment has a segment name and a class name. The linker loads all segments into memory by class name from the first segment encountered to the last. All segments assigned to the same class are loaded into memory contiguously.

During processing, the linker references segments by their addresses in memory (where they are located). The linker does this by finding groups of segments.

A group is a collection of segments that fit within a 64 Kbyte area of memory. The segments do not need to be contiguous to form a group (see Figure 9-2). The address of any group is the lowest address of the segments in that group. At link time, the linker analyzes the groups, then references the segments by the address in memory of that group. A program may consist of one or more groups.

If you are writing in assembly language, you may assign the group and class names in your program. In high-level languages (BASIC, COBOL, FORTRAN, Pascal), the naming is done automatically by the compiler.

Refer to your macro assembler manual for information on how to assign group and class names and on how the linker combines and arranges segments in memory.

FILES THAT THE LINKER USES

The linker:

Works with one or more input files

Produces two output files

May create a temporary disk file

May be directed to search up to eight library files

For each type of file, you may specify a three-part file specification in the format:

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

or a full pathname in the format:

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

Refer to Chapter 3 for a detailed discussion of the format of disk file names.

Input File Extensions

If no filename extensions are given in the input (object) file specifications, the linker recognizes the following extensions by default:

.OBJ	Object
.LIB	Library

Output File Extensions

The linker appends the following default extensions to the output (Run and List) files:

.EXE	Run	(may not be overridden)
.MAP	List	(may be overridden)

VM.TMP (Temporary) File

The linker uses available memory for the link session. If the files to be linked create an output file that exceeds available memory, the linker creates a temporary file named VM.TMP on the default drive. If the linker creates VM.TMP, the following message is displayed:

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

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

Unexpected end of file on VM.TMP

The contents of VM.TMP are written to the file named following the Run File: prompt. VM.TMP is a working file only and is deleted at the end of the linking session.

WARNING! Do not use VM.TMP as a filename for any file. If you have a file named VM.TMP on the default drive and the linker requires the VM.TMP file, the linker deletes the VM.TMP already on disk and create a new VM.TMP. Thus, the contents of the previous VM.TMP file are lost.

HOW TO START THE LINKER

The linker requires two types of input: a command to start the linker and responses to command prompts. In addition, seven parameters control linker features. Usually, you enter all the commands to the linker from the keyboard. As an option, answers to the command prompts and any parameters may be contained in a response file. Command characters can be used to assist you while giving commands to the linker.

The linker may be started in one of three ways. The first method is to start the linker and enter the filenames in response to individual prompts. In the second method, you enter all the filenames on the command line used to start the linker. In the third method, you must create a response file that contains all the necessary filenames and tell the linker where to find the file.

Summary of Starting Methods

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

Method 1: Prompts

To use with Method 1, enter:

LINK<CR>

The linker is loaded into memory and displays a series of four text prompts. The answers to these prompts are the filenames used in the link operation.

One or more parameters may be entered after the filenames on each line to request special linker features.

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

Table 9-1
Linker Command Prompts

Prompt	Responses
Object Modules [.OBJ]:	List .OBJ files to be linked. They must be separated by blank spaces or plus signs (+). If a plus sign is the last character entered, the prompt reappears. There is no default; a response is required.
Run File [Object-file.EXE]:	Enter a filename for the executable object code. The default is first-object-filename.EXE. (You cannot change the output extension.)
List File [NUL.MAP]:	Enter a filename for the listing file. The default is no listing file.
Libraries [.LIB]:	List the filenames to be searched, separated by blank spaces or plus signs (+). If a plus sign is the last character entered, the prompt reappears. The default is to search for default libraries in the object modules. (Extensions are changed to .LIB.)

Method 2: Command Line

To use Method 2, enter all of the filenames on the command line. The entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas. Use the following format:

LINK <object-list>,<runfile>,<listfile>,<lib-list>[/parameter...]

where: <object-list> is a list of object modules, separated by plus signs (+) or spaces.

<runfile> is the name of the file to receive the executable output.

<listfile> is the name of the file to receive the listing.

<lib-list> is a list of library modules to be searched, separated by plus signs (+) or spaces.

/parameter refers to optional parameters, which may be placed following any of the response entries (just before any of the commas or after the <lib-list>, as shown).

To select the default for a field, enter a second comma with no spaces between the two commas.

Example:

LINK FUN+TEXT+TABLE+CARE/P/M,,FUNLIST,COBLIB.LIB<CR>

This command causes the linker to be loaded, then the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ are loaded. The linker then pauses (as a result of using the /P parameter). The linker links the object modules when you press any key, and produces a global symbol map (the /M parameter); defaults to FUN.EXE Run file; creates a List file named FUNLIST.MAP; and searches the Library file COBLIB.LIB.

Method 3: Response File

To start the linker with Method 3, enter:

LINK @<filespec><CR>

where: filespec is the name of a response file. A response file contains answers to the linker prompts (shown in Method 1) and may also contain any of the parameters. When naming a response file, the use of filename extensions is optional. Method 3 permits the start command to be entered from the keyboard or within a batch file without requiring you to take any further action.

To use this option, you must create a response file containing several lines of text, each of which is the response to a command prompt. The responses must be in the same order as the command prompts discussed in Method 1. If desired, a long response to the Object Modules: or Libraries: prompt may be entered on several lines by using a plus sign (+) to continue the same response onto the next line.

Use parameters and command characters in the response file the same way as they are used for responses entered at the keyboard.

When the linking session begins, each prompt is displayed in order on the screen with the response from the response file. If the response file does not contain an answer for all the prompts (in the form of filenames, the semicolon command character or carriage returns), the command prompts without responses are displayed and the linker waits for a legal response from the keyboard. When a legal response has been entered, the linker continues the link session.

Example:

```
FUN TEXT TABLE CARE
/PAUSE/MAP
FUNLIST
COBLIB.LIB
```

This response file tells the linker to load the four object modules named FUN, TEXT, TABLE, and CARE. The linker pauses before producing a public symbol map to permit you to swap disks (see discussion under /PAUSE in the "Parameters" section before using this feature). When you press any key, the output files are named FUN.EXE and FUNLIST.MAP. The linker searches the library file COBLIB.LIB.

COMMAND CHARACTERS

The linker provides the three command characters listed in Table 9-2.

Table 9-2
Linker Command Characters

Character	Function
(+)	Use the plus sign (+) to separate entries and to extend the current line in response to the Object Modules: and Libraries: prompts. (A blank space may be used to separate object modules.) To enter a large number of responses (each may be very long), enter a plus sign and <CR> at the end of the line to extend it. If the plus sign and <CR> are the last entries following these two prompts, the linker prompts you for more module names. When the Object Modules: or Libraries: prompt appears again, continue to enter responses. When all the modules to be linked and libraries to be searched have been listed, be sure the response line ends with a module name and a <CR> and not a plus sign and <CR>. Example: Object Modules [.OBJ]: **FUN TEXT TABLE CARE+<CR>** Object Modules [.OBJ]: **FOO+FLIPFLOP+JUNQUE+<CR>** Object Modules [.OBJ]: **CORSAIR<CR>**
(;)	To select default responses to the remaining prompts, use a single semicolon (;) followed immediately by a carriage return at any time after the first prompt (Run File:). This feature saves time and overrides the need to press a series of <CR> keys.

Table 9-2 (Continued)
Linker Command Characters

Character Function

NOTE! Once the semicolon has been entered, you can no longer respond to any of the prompts for that link session. Therefore, do not use the semicolon to skip some prompts. To skip prompts, press <CR>.

Example:

Object Modules [.OBJ]: **FUN TEXT TABLE CARE<CR>**
Run Module [FUN.EXE]: **;<CR>**

No other prompts appear, and the linker uses the default values (including FUN.MAP for the List file).

<^C> Use the <Ctrl>/<C> or <^C> key sequence to abort the link session at any time. If you enter an erroneous response, such as the wrong filename or an incorrectly spelled filename, you must press <^C> to exit the link program, and then restart the linker. If the error has been entered but you have not pressed <CR>, you may delete the erroneous characters with the backspace key, but for that line only.

COMMAND PROMPTS

The linker asks you for responses to four text prompts. When you have entered a response to a prompt and pressed <CR>, the next prompt appears. When the last prompt has been answered, the linker begins linking automatically without further commands. When the link session is finished, the linker exits to the operating system. If the link session is unsuccessful, the linker displays the appropriate error message.

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

Object Modules [.OBJ]:

Enter a list of the object modules to be linked. The linker assumes by default that the filename extension is .OBJ. If an object module has any other filename extension, the extension must be given.

Modules may be separated by spaces or plus signs (+).

Remember that the linker loads segments into classes in the order encountered. You can use this information to set the order in which the object modules are read.

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

Enter a filename for the Run (executable) file that results from the link session. All Run files receive the filename extension .EXE, even if you specify an extension other than .EXE.

If no response is entered to the Run File: prompt, the first filename entered in response to the Object Modules: prompt is used as the RUN filename.

Example:

Run File [FUN.EXE]: **B:PAYROLL<CR>**

This response creates the Run file PAYROLL.EXE on drive B.

List File [NUL.MAP]:

The List file contains an entry for each segment in the input (object) modules. Each entry also shows the addressing in the Run file.

The default response is no list file.

Libraries [.LIB]:

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

Library filenames must be separated by blank spaces or plus signs (+).

Library files are searched in the order listed to resolve external references. When the module that defines the external symbol is found, that module is processed as another object module.

If the library file cannot be found on the disks in the disk drives, the following message is displayed:

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

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

LINKER PARAMETERS

Seven parameters are used to control various linker functions. Parameters must be entered at the end of a prompt response, regardless of which method is used to start the linker. Parameters may be grouped at the end of any response, or may be scattered at the end of several. If more than one parameter is entered at the end of one response, each parameter must be preceded by a forward slash (/).

All parameters may be abbreviated. The only restriction is that an abbreviation must be sequential from the first letter through the last entered; no gaps or transpositions are allowed. For example:

Legal	Illegal
/D	/DSL
/DS	/DAL
/DSA	/DLC
/DSALLOCA	/DSALLOCT

Table 9-3
Linker Parameters

Parameter	Function
/DSALLOCATE	Using the /DSALLOCATE parameter tells the linker to load all data at the high end of the Data Segment. Otherwise, all data is loaded at the low end of the Data Segment. At runtime, the DS pointer is set to the lowest possible address to allow the entire DS segment to be used. Use of the /DSALLOCATE parameter in combination with the default load low (that is, the /HIGH parameter is not used) permits the user application to dynamically allocate any available memory below the area specifically allocated within DGROUP, yet to remain addressable by the same DS pointer. This dynamic allocation is needed for Pascal and FORTRAN programs.

Table 9-3 (Continued)
Linker Parameters

Parameter	Function
	NOTE! Your applications program may dynamically allocate up to 64 Kbytes (or the actual amount of memory available) less the amount allocated within DGROUP.
/HIGH	Use the /HIGH parameter to place the Run file as high as possible in memory. Otherwise, the Run file is placed as low as possible.
	NOTE! Do not use the /HIGH parameter with Pascal or FORTRAN programs.
/LINENUMBERS	The /LINENUMBERS parameter requests the linker to include in the List file the line numbers and addresses of the source statements in the input modules. Otherwise, line numbers are not included in the List file.
	NOTE! Not all compilers produce object modules that contain line number information. In these cases, of course, the linker cannot include line numbers.
/MAP	/MAP directs the linker to list all public (global) symbols defined in the input modules. If /MAP is not given, only errors are listed (including undefined globals).
	The symbols are listed alphabetically. For each symbol, the value and the segment:offset location in the Run file are listed. The symbols are listed at the end of the List file.
/PAUSE	The /PAUSE parameter requests the linker to pause in the link session when the parameter is encountered. Normally the link session is performed from beginning to end without stopping. This parameter allows you to swap diskettes before the Run (.EXE) file is output.

Table 9-3 (Continued)
Linker Parameters

Parameter	Function
	When the /PAUSE parameter is encountered, the following message is displayed: About to generate .EXE file Change disks <hit any key> processing resumes when a key is pressed. **STOP! Do not remove the disk which will receive the List file, or the disk used for the VM.TMP file, if one has been created.**
/STACK:<number>	<number> represents any positive numeric value (in hexadecimal) up to 65536 bytes. If a value from 1 to 511 is entered, 512 is used. If the /STACK parameter is not used for a link session, the linker calculates the necessary stack size automatically. All compilers and assemblers should provide information in the object modules that allow the linker to compute the required stack size. At least one object (input) module must contain a stack allocation statement. If not, the linker displays the following error message: WARNING: NO STACK STATEMENT
/NO	/NO is short for NO DEFAULT LIBRARY SEARCH. This parameter tells the linker not to search the default (product) libraries in the object modules. For example, if you are linking object modules in Pascal, specifying the /NO parameter tells the linker not to automatically search the library named PASCAL.LIB to resolve external references.

SAMPLE LINK SESSION

This sample shows you the type of information that is displayed during a link session.

In response to the TeleDOS prompt, enter:

LINK<CR>

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

```
Microsoft Object Linker V.2.00
(C) Copyright 1982 by Microsoft Inc.

Object Modules [.OBJ]: IO SYSINIT<CR>
Run File [IO.EXE]: <CR>
List File [NUL.MAP]: IO /MAP<CR>
Libraries [.LIB]: <CR>
```

Notes:

1. By specifying /MAP, you get both an alphabetic listing and a chronological listing of public symbols.

2. By pressing <CR> in response to the Libraries: prompt, an automatic library search is performed.

Once all the libraries are located, the linker map displays a list of segments in the order of their appearance within the load module. The list might look like this:

Start	Stop	Length	Name
00000H	009ECH	09EDH	CODE
009F0H	01166H	0777H	SYSINITSEG

The information in the Start and Stop columns shows the 20-bit hex address of each segment relative to location zero. Location zero is the beginning of the load module. The addresses displayed are not the absolute addresses where these segments are loaded.

Because the /MAP parameter was used, the public symbols are listed by name and value. For example:

ADDRESS	PUBLICS_BY_NAME
009F:0012	BUFFERS
009F:0005	CURRENT_DOS_LOCATION
009F:0011	DEFAULT_DRIVE
009F:000B	DEVICE_LIST
009F:0013	FILES
009F:0009	FINAL_DOS_LOCATION
009F:000F	MEMORY_SIZE
009F:0000	SYSINIT

ADDRESS	PUBLICS BY VALUE
009F:0000	SYSINIT
009F:0005	CURRENT_DOS_LOCATION
009F:0009	FINAL_DOS_LOCATION
009F:000B	DEVICE_LIST
009F:000F	MEMORY_SIZE
009F:0011	DEFAULT_DRIVE
009F:0012	BUFFERS
009F:0013	FILES

ERROR MESSAGES

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

Attempt to access data outside of segment bounds

There is probably an invalid Object file.

Bad numeric parameter

The value entered with the /STACK parameter is not a valid numeric constant.

Cannot nest response file

A @<response file> was used within an automatic response file. Response files cannot be nested.

Cannot open list file

The directory or disk is full.

Cannot open overlay

The directory or disk is full.

Cannot open response file

The automatic response file could not be found.

Cannot open temporary file

The linker is unable to create the file VM.TMP because the disk directory is full. Insert a new disk. Do not remove the disk that will receive the List.MAP file.

DUP record too complex

The DUP record in an assembly language module is too complex. Simplify the DUP record in the assembly language program.

Fixup offset exceeds field width

An assembly language instruction refers to an address with a short instruction instead of a long instruction. Edit the assembly language source and reassemble.

Invalid object module

An object module(s) is incorrectly formed or incomplete (as when assembly is stopped in the middle).

Out of space on list file

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

Out of space on run file

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

Out of space on VM.TMP

No more disk space remains to expand the VM.TMP file.

Program size exceeds capacity of Link

The total size may not exceed 384 Kbytes and the number of segments may not exceed 255.

Requested stack size exceeds 64K

Specify a size greater than or equal to 64 Kbytes with the /STACK parameter.

Segment size exceeds 64K

64 Kbytes is the addressing system limit.

Stack size exceeds 65535 bytes

The size specified for the stack must be less than or equal to 65535.

Symbol defined more than once

The linker found two or more modules that define a single symbol name.

Symbol table capacity exceeded

Very many and/or very long names were entered, exceeding the limit of approximately 25 Kbytes.

Too many external symbols in one module

The limit is 256 external symbols per module.

Too many groups

The limit is ten groups.

Too many libraries specified

The limit is eight libraries.

Too many overlays

The is a limit of 64 overlays.

Too many public symbols

The limit is 1024 public symbols.

Too many segments or classes

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

Unresolved externals: <list>

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

Warning: no STACK segment

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

10. DEBUG Utility

OVERVIEW OF DEBUG

The DEBUG Utility (DEBUG) is a debugging program that provides a controlled testing environment for binary and executable object files. Note that EDLIN is used to alter source files; DEBUG is EDLIN's counterpart for binary files. DEBUG eliminates the need to reassemble a program to see if a problem has been fixed by a minor change. It allows you to alter the contents of a file or the contents of a CPU register, and then to immediately reexecute a program to check on the validity of the changes.

All DEBUG commands may be aborted at any time by pressing <Ctrl>/<Break>. <Ctrl>/<Num Lock suspends the display, so that you can read it before the output scrolls away. Entering any key other than <Ctrl>/<Break> or <Ctrl>/<Num Lock> restarts the display.

HOW TO START DEBUG

DEBUG may be started two ways. In the first method, you start DEBUG and then enter all commands in response to the DEBUG prompt (a hyphen). In the second method, you enter all commands on the command line used to start DEBUG.

Summary of Methods to Start DEBUG

Method 1	**DEBUG<CR>**
Method 2	**DEBUG [<pathname> [<arglist>]]<CR>**

Method 1: DEBUG

To start DEBUG using method 1, enter:

DEBUG<CR>

DEBUG responds with the hyphen (-) prompt, signaling that it is ready to accept your commands. Since no filename has been specified, current memory, disk sectors, or disk files can be worked on by using other commands.

Warnings

1. When DEBUG (Version 2.0) is started, it sets up a program header at offset 0 in the program work area. You can overwrite the default header if no <pathname> is given to DEBUG. If you are debugging a .COM or .EXE file, however, do not tamper with the program header below address 5CH, or DEBUG will terminate.

2. Do not restart a program after the Program terminated normally message is displayed. You must reload the program with the N and L commands for it to run properly.

Method 2: Command Line

To start DEBUG using a command line, enter:

DEBUG [<pathname> [<arglist>]]<CR>

For example, if a <pathname> is specified, then the following is a typical command to start DEBUG:

DEBUG FILE.EXE<CR>

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.

An <arglist> may be specified if <pathname> is present. The <arglist> is a list of filename parameters that are to be passed to the program <pathname>. Thus, when <pathname> is loaded into memory, it is loaded as if it had been started with the command:

<pathname> <arglist>

Here, <pathname> is the file to be debugged, and the <arglist> is the rest of the command line that is used when <pathname> is invoked and loaded into memory.

PARAMETERS

All DEBUG commands accept parameters, except the Quit command. Parameters may be separated by delimiters (spaces or commas), but a delimiter is required only between two consecutive hexadecimal values. Thus, the following commands are equivalent:

```
DCS:100 110
D CS:100 110
D,CS:100,110
```

Table 10-1
DEBUG Command Parameters

PARAMETER	DEFINITION
<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 04BA:0100 **The colon is required between a segment designation (whether numeric or alphabetic) and an offset.**
<byte>	A two-digit hexadecimal value to be placed in or read from an address or register.
<drive>	A one-digit hexadecimal value to indicate which drive a file is 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:.
<list>	A series of <byte> values or of <string>s. <list> must be the last parameter on the command line. Example: FCS:100 L8 42 45 52 54 41 'ABC'

Table 10-1 (Continued)
DEBUG Command Parameters

PARAMETER	DEFINITION
<range>	An address range specifying a lower and upper address. Two formats can be used: <address> <address> or <address> L <value> where <value> is the number of bytes in hex to be processed. Example: CS:100 110 CS:100 L 10 The following is illegal: CS:100 CS:110 ^ Error The limit for <range> is 10000 hex. To specify a <value> of 10000 hex within four digits, enter 0000 (or 0).
<sector> <sector>	1- to 3-digit hexadecimal values used to indicate the starting relative sector on the disk and the number of disk sectors to be written or loaded. Relative sectors are obtained by counting the sectors on the disk surface. The first sector (relative sector 0) is at track 0, sector 1, head 0. Numbering continues for each sector on that track and head, and then continues with the first sector on the next head of the same track. When all the sectors on all heads of that track have been counted, numbering continues with the first sector on head 0 of the next track.
<string>	Any number of characters enclosed in quote marks. Quote marks may be either single (') or double("). If the delimiter quote marks must appear within a <string>, the quote marks must be doubled. For example, the following strings are legal: 'This is a "string" is okay.' 'This is a ''string'' is okay.' However, this string is illegal: 'This is a 'string' is not.'

Table 10-1 (Continued)
DEBUG Command Parameters

PARAMETER	DEFINITION
	Similarly, these strings are legal: "This is a 'string' is okay." "This is a ""string"" is okay." However, this string is illegal: "This is a "string" is not." Note that the double quote marks are not necessary in the following strings: "This is a ''string'' is not necessary." 'This is a ""string"" is not necessary.' The ASCII values of the characters in the string are used as a <list> of byte values.
<value>	A hexadecimal value up to four digits used to specify a port number, numbers to be added or subtracted (Hexarithmetic command), number of bytes, or the number of times a command should repeat its functions.

COMMANDS

Each DEBUG command consists of a single letter followed by parameters. Additionally, the control characters and the special editing functions described in Chapter 6 apply inside DEBUG.

If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with an up-arrow (^) and the word Error."

For example:

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

Any combination of upper-case and lower-case letters may be used in commands and parameters.

The DEBUG commands are summarized in Table 10-2 and are described in detail on the following pages.

Table 10-2
DEBUG Commands

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

A (Assemble) Command

Purpose Assembles 8086/8087/8088 mnemonics directly into memory.

Format A[<address>]

Comments If a syntax error is found, DEBUG responds with

```
^Error
```

and redisplays the current assembly address.

All numeric values are hexadecimal and must be 1-4 characters. Prefix mnemonics must be specified in front of the opcode to which they refer. They may also be entered on a separate line.

The segment override mnemonics are CS:, DS:, ES:, and SS:. The mnemonic for the far return is RETF. String manipulation mnemonics must explicitly state the string size. For example, use MOVSW to move word strings and MOVSB to move byte strings.

The assembler automatically assembles short, near, or far jumps and calls, depending on byte displacement to the destination address. These may be overridden with the NEAR or FAR prefix. For example:

```
0100:0500 JMP   502          ; a 2-byte short jump
0100:0502 JMP   NEAR 505     ; a 3-byte near jump
0100:0505 JMP   FAR  50A     ; a 5-byte far jump
```

The NEAR prefix may be abbreviated to NE, but the FAR prefix cannot be abbreviated.

DEBUG cannot tell whether some operands refer to a word memory location or to a byte memory location. In this case, the data type must be explicitly stated with the prefix "WORD PTR" or "BYTE PTR". Acceptable abbreviations are "WO" and "BY". For example:

```
NEG     BYTE PTR [128]
DEC     WO [SI]
```

A (Assemble) Command

DEBUG also cannot tell whether an operand refers to a memory location or to an immediate operand. DEBUG uses the common convention that operands enclosed in square brackets refer to memory. For example:

```
MOV     AX,21          ; Load AX with 21H
MOV     AX,[21]        ; Load AX with the
                       ; contents of memory
                       ; location 21H
```

Two popular pseudo-instructions are available with Assemble. The DB opcode assembles byte values directly into memory. The DW opcode assembles word values directly into memory. For example:

```
DB      1,2,3,4,"THIS IS AN EXAMPLE"
DB      'THIS IS A QUOTE: "'
DB      "THIS IS A QUOTE: '"
DW      1000,2000,3000,"BACH"
```

Assemble supports all forms of register indirect commands. For example:

```
ADD     BX,34[BP+2].[SI-1]
POP     [BP+DI]
PUSH    [SI]
```

All opcode synonyms are also supported. For example:

```
LOOPZ   100
LOOPE   100
JA      200
JNBE    200
```

For 8087 opcodes, the WAIT or FWAIT must be explicitly specified. For example:

```
FWAIT FADD ST,ST(3)    ; This line will assemble
                       ; a FWAIT prefix
LD TBYTE PTR [BX]      ; This line will not
```

C (Compare) Command

Purpose Compares the portion of memory specified by <range> to a portion of the same size beginning at <address>.

Format C<range> <address>

Comments If the two areas of memory are identical, there is no display and DEBUG returns with the (-) prompt. If there are differences, they are displayed in this format:

<address1> <byte1> <byte2> <address2>

If only an offset is entered for <range>, the C command assumes the segment contained in the DS register.

Examples The following commands have the same effect:

C100,1FF 300<CR>

or

C100L100 300<CR>

Each command compares the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH. The segment in the DS register is used by default.

D (Dump) Command

Purpose Displays the contents of the specified region of memory.

Format D[<range>]

Comments If a range of addresses is specified, the contents of the range are displayed. If the D command is entered without parameters, 128 bytes are displayed at the first address (DS:100) after the address displayed by the previous Dump command.

The dump is displayed in two portions: a hexadecimal dump (each byte is shown in hexadecimal value) and an ASCII dump (the bytes are shown in ASCII characters). Non-printing characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows 16 bytes with a hyphen between the eighth and ninth bytes. At times, displays are split in this manual to fit them on the page. Each displayed line begins on a 16-byte boundary.

Examples If you enter the command:

DCS:100 10F<CR>

DEBUG displays the sixteen memory locations from CS:100 to CS:10F in the following format.

```
04BA:0100 43 61 6E 6E ... 73 70  Cannot open resp
```

If you enter the following command:

D<CR>

the next 128 bytes are displayed. 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 128 bytes immediately following those last displayed.

E (Enter) Command

Purpose Enters byte values into memory at the specified <address>.

Format E<address> [<list>]

Comments If the optional <list> of values is entered, the replacement of byte values occurs automatically. (If an error occurs, no byte values are changed.)

If the <address> is entered without the optional <list>, DEBUG displays the address and its contents, then repeats the address on the next line and waits for your input. At this point, the Enter command waits for you to perform one of the following actions:

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

2. Press the <Space Bar> to advance to the next byte. To change the value, enter the new value as described in (1.) above. If you space beyond an 8-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.

3. Enter hyphen (-) to return to the preceding byte. If you decide to change a byte behind the current position, entering the hyphen returns the current position to the previous byte. When the hyphen is entered, a new line is started with the address and its byte value displayed.

4. Press <CR> to terminate the Enter command. The <CR> key may be pressed at any byte position.

Example Assume that the following command is entered:

ECS:100<CR>

Suppose DEBUG displays:

04BA:0100 EB._

To change this value to 41, enter 41 as shown:

04BA:0100 EB.**41**_

E (Enter) Command

To step through the subsequent bytes, press the <Space Bar> three times to see:

```
04BA:0100  EB.41    10.    00.    BC._
```

To change BC to 42, enter 42 as shown:

```
04BA:0100  EB.41    10.    00.    BC.42_
```

Now, realizing that 10 should be 6F, enter the hyphen twice to return to byte 0101 (value 10), then replace 10 with 6F:

```
04BA:0100  EB.41    10.    00.    BC.42-
04BA:0102  00.-_
04BA:0101  10.6F_
```

Pressing <CR> changes any entered values in memory, ends the Enter command, and returns to the DEBUG command level.

F (Fill) Command

Purpose Fills the addresses in the <range> with the values in the <list>.

Format F<range> <list>

Comments 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 any of the memory in the <range> is not valid (bad or nonexistent), an error occurs in all succeeding locations.

Example Assume that the following command is entered:

F04BA:100 L 100 42 45 52 54 41<CR>

DEBUG fills memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values are repeated until all 100H bytes are filled.

G (Go) Command

Purpose Executes the program currently in memory.

Format G[=<address> [<address>...]]

Comments If only the Go command is entered, the program executes as if the program had run outside DEBUG.

If =<address> is set, execution begins at the address specified. The equal sign (=) is required, so that DEBUG can distinguish the start =<address> from the breakpoint <address>es.

With the other optional addresses set, execution stops at the first <address> encountered, regardless of that address' position in the list of addresses to halt execution or program branching. When program execution reaches a breakpoint, the registers, flags, and decoded instruction are displayed for the last instruction executed. (The result is the same as if you had entered the Register command for the breakpoint address.)

Up to ten breakpoints may be set. Breakpoints may be set only at addresses containing the first byte of an 8086 opcode. If more than ten breakpoints are set, DEBUG returns the BP error message (see the error message listing at the end of this chapter).

The user stack pointer must be valid and have 6 bytes available for this command. The G command uses an IRET instruction to cause a jump to the program under test. The user stack pointer is set, and the user flags, Code Segment register, and Instruction Pointer are pushed on the user stack. (Thus, if the user stack is not valid or is too small, the operating system may crash.) An interrupt code (0CCH) is placed at the specified breakpoint address(es).

When an instruction with the breakpoint code is encountered, all breakpoint addresses are restored to their original instructions. If execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

G (Go) Command

Example Assume that the following command is entered:

GCS:7550<CR>

The program currently in memory begins with the current instruction (current address of CS:IP) and executes up to the address 7550 in the CS segment. DEBUG then displays registers and flags, after which the Go command is terminated.

After a breakpoint has been encountered, if you enter the Go command again, the program executes just as if you had entered the filename at the TeleDOS command level. The only difference is that program execution begins at the instruction after the breakpoint rather than at the usual start address.

H (Hex) Command

Purpose Performs hexadecimal arithmetic on the two values specified.

Format H<value> <value>

Comments First, DEBUG adds the two parameters, then subtracts the second parameter from the first. The results of the arithmetic are displayed on one line; first the sum, then the difference.

Example Assume that the following command is entered:

H19F 10A<CR>

DEBUG performs the calculations and then displays the result:

02A9 0095

I (Input) Command

Purpose Inputs and displays one byte from the port specified by <value>.

Format I<value>

Comments A 16-bit port address is allowed.

Example Assume that you enter the following command:

I2F8<CR>

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

42

L (Load) Command

Purpose Loads a file into memory.

Format L[<address> [<drive> <sector> <sector>]]

Comments If only the L command is entered (no parameters), DEBUG loads into memory starting at location CS:100 the file whose filespec is in the file control block at CS:5C. A file is placed in the file control block by entering the filespec on the command line when DEBUG is started, or by using the NAME command. The L command sets the BX and CX registers to the number of bytes that have been loaded into memory.

If the L command is entered with only the <address> parameter, the file specified in the file control block is loaded into memory at the specified address.

Files with a file extension of .COM are always loaded into memory at location CS:100, regardless of a specified <address> in the L command.

If the file has a .EXE extension, it is relocated to the load address specified in the header of the .EXE file: the <address> parameter is always ignored for .EXE files. The header itself is stripped off the .EXE file before it is loaded into memory. Thus the size of an .EXE file on disk differs from its size in memory.

If the file named by the Name command or specified when DEBUG is started is a .HEX file, then entering the L command with no parameters causes DEBUG to load the file beginning at the address specified in the .HEX file. If the L command includes the option <address>, DEBUG adds the <address> specified in the L command to the address found in the .HEX file to determine the start address for loading the file.

If L is entered with all the parameters, absolute disk sectors are loaded into memory. The sectors are taken from the <drive> specified (the drive designation is numeric where 0=A:, 1=B:, 2=C:, and 3=D:); DEBUG begins loading with the first <sector> specified, and continues until the number of sectors specified in the second <sector> have been loaded.

The maximum number of sectors that can be loaded by an L command is hex 80.

L (Load) Command

Example Assume that the following commands are entered:

```
A>DEBUG<CR>
-NFILE.COM
```

Now, to load FILE.COM, enter:

```
L<CR>
```

FILE.COM is loaded into memory at CS:100 and DEBUG returns the (-) prompt.

The command:

```
LCS:100 1 0F 6D<CR>
```

would load 109 (6D hex) sectors from drive B beginning with logical sector 15 (0F hex) into memory starting at address CS:0100. When the records have been loaded, DEBUG returns the (-) prompt.

M (Move) Command

Purpose Moves the block of memory specified by <range> to the location beginning at the <address> specified.

Format M<range> <address>

Comments 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. The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block's lowest address and then to work towards the highest. The sequence for moves from lower addresses to higher addresses is to move the data beginning at the block's highest address and to work towards the lowest.

Note that if the addresses in the block being moved do not have new data moved to them, the data there before the move remains. The M command copies the data from one area into another, in the sequence described, and writes over the new addresses. This is why the sequence of the move is important.

If only an offset is entered for the starting address of the <range> parameter or for the <address> parameter, the M command uses the segment contained in the DS register.

Example Assume that you enter:

MCS:100 110 CS:500<CR>

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 M command, to review the results of the move.

N (Name)
Command

Purpose Sets filenames.

Format N<filespec> [<filespec>...]

Comments The N command performs two functions. First, N is used to assign a filename for a later Load or Write command. If you start DEBUG without naming any file to be debugged, the N<filespec> command must be entered before a file can be loaded. Second, N is used to assign filename parameters to the file being debugged. In this case, Name accepts a list of parameters that are used by the file being debugged.

These two functions overlap. Consider the following set of DEBUG commands:

```
-NFILE1.EXE<CR>
-L<CR>
-G<CR>
```

Because of the effects of the N command, Name performs the following steps:

1. (N)ame assigns the filename FILE1.EXE to the file to be used in any later Load or Write commands.

2. (N)ame also assigns the filename FILE1.EXE to the first filename parameter used by any program that is later debugged.

3. (L)oad loads FILE1.EXE into memory.

4. (G)o causes FILE1.EXE to be executed with FILE1.EXE as the single filename parameter (that is, FILE1.EXE is executed as if FILE1.EXE had been entered at the command level).

A more useful chain of commands might look like this:

```
-NFILE1.EXE<CR>
-L<CR>
-NFILE2.DAT FILE3.DAT<CR>
-G<CR>
```

N (Name)
Command

Here, Name sets FILE1.EXE as the filename for the subsequent Load command. The Load command loads FILE1.EXE into memory, and then the N 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 entered at the TeleDOS command level. Note that if a Write command were executed at this point, FILE1.EXE (the file being debugged) would be saved with the name FILE2.DAT! To avoid such undesired results, you should always execute a N command before either a Load or a Write.

There are four regions of memory that can be affected by the N command:

CS:5C	FCB for file 1
CS:6C	FCB for file 2
CS:80	Count of characters
CS:81	All characters entered

A File Control Block (FCB) for the first filename parameter given to the N command is set up at CS:5C. If a second filename parameter is entered, then an FCB is set up for it beginning at CS:6C. The number of characters entered in the N command (exclusive of the first character N) is given at location CS:80. The actual stream of characters given by the N command (again, exclusive of the character N) begins at CS:81. Note that this stream of characters may contain parameters and delimiters that would be legal in any command entered at the TeleDOS command level.

Example A typical use of the N command is:

```
A>DEBUG PROG.COM<CR>
-NPARAM1 PARAM2/C<CR>
-G<CR>
```

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

```
PROG PARAM1 PARAM2/C<CR>
```

Testing and debugging therefore reflect a normal runtime environment for PROG.COM.

O (Output) Command

Purpose Sends the <byte> specified to the output port specified by <value>.

Format O<value> <byte>

Comments A 16-bit port address is allowed.

Example Enter:

O2F8 4F<CR>

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

Q (Quit) Command

Purpose Terminates the DEBUG utility.

Format Q

Comments The Q command takes no parameters and exits DEBUG without saving the file currently being operated on. You are returned to the TeleDOS command level.

Example To end the debugging session, enter:

Q<CR>

DEBUG is terminated and control returns to the TeleDOS command level.

R (Register) Command

Purpose Displays the contents of one or more CPU registers.

Format R[<register-name>]

Comments If no <register-name> is entered, the R command dumps the register save area and 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 then either enter a <value> to change the register, or press <CR> if no change is wanted.

The only valid <register-name>s are:

AX	BP	SS	
BX	SI	CS	
CX	DI	IP	(IP and PC both refer to the
DX	DS	PC	Instruction Pointer).
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 each flag with a two-character alphabetic code. To alter any flag, enter the opposite two-letter code. The flags are either set or cleared.

The flags are listed below with their codes for SET and CLEAR:

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

R (Register) Command

Whenever you enter the command RF, the flags are displayed in the order shown above in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a hyphen (-). You may enter new flag values as alphabetic pairs. The new flag values can be entered in any order. You do not have to leave spaces between the flag entries. To exit the R command, press <CR>. Flags for which new values were not entered remain unchanged.

If more than one value is entered for a flag, DEBUG returns a DF Error message. If you enter a flag code other than those shown above, DEBUG returns a BF Error message. In both cases, the flags up to the error in the list are changed; flags at and after the error are not.

At startup, the segment registers are set to the bottom of free memory, the Instruction Pointer is set to 0100H, all flags are cleared, and the remaining registers are set to zero.

Example Enter:

R<CR>

DEBUG displays all registers, flags, and the decoded instruction for the current location. If the location is CS:11A, then the display will look similar to this:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000
SI=005C DI=0000 DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A  NV UP DI NG NZ AC PE NC
04BA:011A  CD21          INT     21
```

If you enter:

RF<CR>

DEBUG might display the following flag settings:

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

R (Register)
Command

To change the settings, enter any valid flag designation, in any order, with or without spaces.

For example:

NV UP DI NG NZ AC PE NC - **PLEICY<CR>**

DEBUG responds only with the DEBUG prompt. To see the changes, enter either the R or RF command:

RF<CR>

DEBUG displays the new settings:

NV UP EI PL NZ AC PE CY - _

Press <CR> to leave the flags this way.

S (Search) Command

Purpose Searches the <range> specified for the <list> of bytes specified.

Format S<range> <list>

Comments The <list> may contain one or more bytes, each separated by a space or comma. If the <list> contains more than one byte, only the first address of the byte string is returned. If the <list> contains only one byte, all addresses of the byte in the <range> are displayed.

Example If you enter:

SCS:100 110 41<CR>

DEBUG displays a response similar to this:

```
04BA:0104
04BA:010D
```

T (Trace) Command

Purpose Executes one instruction and displays the contents of all registers and flags, and the decoded instruction.

Format T[=<address>] [<value>]

Comments If the optional =<address> is entered, tracing occurs at the =<address> specified. The optional <value> causes DEBUG to execute and trace the number of steps specified by <value>.

The T command uses the hardware trace mode of the 8086 or 8088 microprocessor. Consequently, you may also trace instructions stored in ROM (Read Only Memory).

Example Enter:

T<CR>

DEBUG returns a display of the registers, flags, and decoded instruction for that one instruction. Assume that the current position is 04BA:011A; DEBUG might return the display:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000
SI=005C DI=0000 DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A  NV UP DI NG NZ AC PE NC
04BA:011A  CD21          INT     21
```

If you enter:

T=011A 10<CR>

DEBUG executes sixteen (10 hex) instructions beginning at 011A in the current segment, and then displays all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed. Then the display stops, and you can see the register and flag values for the last few instructions performed. Remember that <Ctrl>/<Num Lock> suspends the display at any point, so that you can study the registers and flags for any instruction.

U (Unassemble) Command

Purpose Disassembles bytes and displays the source statements that correspond to them, with addresses and byte values.

Format U[<range>]

Comments The display of disassembled code looks like a listing for an assembled file. If you enter the U command without parameters, 20 hexadecimal bytes are disassembled at the first address after that displayed by the previous Unassemble command. If you enter the U command with the <range> parameter, DEBUG disassembles all bytes in the range. If the <range> is given as an <address> only, then 20H bytes are disassembled starting at that <address>.

Example Enter:

U04BA:100 L10<CR>

DEBUG disassembles 16 bytes beginning at address 04BA:0100 in the following format:

```
04BA:0100  206472    AND  [SI+72],AH
04BA:0103  69        DB   69
04BA:0104  7665      JBE  016B
04BA:0106  207370    AND  [BP+DI+70],DH
04BA:0109  65        DB   65
04BA:010A  63        DB   63
04BA:010B  69        DB   69
04BA:010C  66        DB   66
04BA:010D  69        DB   69
04BA:010E  63        DB   63
04BA:010F  61        DB   61
```

If you enter:

U04BA:0100 0108<CR>

The display shows:

```
04BA:0100  206472    AND  [SI+72],AH
04BA:0103  69        DB   69
04BA:0104  7665      JBE  016B
04BA:0106  207370    AND  [BP+DI+70],DH
```

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

W (Write) Command

Purpose Writes the file being debugged to a disk file.

Format W[<address> [<drive> <sector> <sector>]]

Comments If you enter the W command with no parameters, the file in memory starting at memory location CS:100 is written to disk using the filespec contained in the file control block at CS:5C. The BX and CX registers must be set to the number of bytes to be written.

If the W command is entered with just the <address> parameter, the file starting at memory location <address> is written to disk using the filespec contained in the file control block at CS:5C. Again, the BX and CX registers must be set to the number of bytes to be written.

Note, if a Go or Trace command has been used, the values placed in the BX and CX registers when the file was loaded might have been changed.

When a file is loaded, edited under DEBUG, and then written back to disk with the same filespec, the edited file is written over the original file.

If the W command is entered with all of the parameters, data is written from memory location <address> to the specified <drive> (the drive designation is numeric, where 0=A:, 1=B:, 2=C:, and 3=D:). DEBUG writes the data to disk beginning at the logical sector number specified by the first <sector> parameter and continues to write until the number of sectors specified in the second <sector> parameter have been written.

WARNING Writing to absolute sectors is EXTREMELY dangerous because the process bypasses the file handler.

W (Write) Command

Examples If you enter:

W<CR>

DEBUG writes the file starting at memory location CS:100 to disk using the filespec contained in the file control block at CS:5C. The number of bytes written to disk are contained in the BX and CX registers. DEBUG then display the DEBUG prompt.

Entering:

WCS:100 1 37 2B<CR>

DEBUG writes the contents of memory beginning at address CS:100 to the disk in drive B:. 2BH sectors of data are written to disk starting at logical sector number 37H.

ERROR MESSAGES

Duringthe DEBUG session,you may receive any of thefollowing errormessages. Each error terminates the DEBUG commandunder which it occurred, but does not terminate DEBUG itself.

Table 10-3
DEBUG Error Codes

ERROR CODE	DEFINITION
BF	Bad flag You attempted to alter a flag, but the characters entered were not one of the acceptable pairs of flag values. 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 G command. Reenter the G command with ten or fewer breakpoints.
BR	Bad register You entered the R 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.

APPENDIX A INSTRUCTIONS FOR USERS WITH SINGLE-DRIVE SYSTEMS

On a single-drive system, you enter the commands as you would on a multi-drive system.

You should think of the single-drive system as having two drives (drive A and drive B). But instead of A and B representing two physical drives as on the multi-drive system, the A and B represent diskettes.

If you specify drive B when the "drive A diskette" was last used, you are prompted to insert the diskette for drive B. For example:

```
A> COPY COMMAND.COM B:<CR>

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

        1 File(s) copied

A>_
```

If you specify drive A when the "drive B diskette" was last used, you are prompted again to change diskettes. This time, TeleDOS prompts you to insert the "drive A diskette."

The same procedure is used if a command is executed from a batch file. TeleDOS waits for you to insert the appropriate diskette and press any key before it continues.

NOTE! The letter displayed in the system prompt represents the default drive where TeleDOS looks to find a file whose name is entered without a drive specifier. The letter in the system prompt does not represent the last diskette used.

For example, assume that A is the default drive. If the last operation performed was DIR B:, TeleDOS believes the "drive B diskette" is still in the drive, even though the system prompt is A> and A drive is still the default drive. If you now enter DIR, TeleDOS prompts you for the "drive A diskette" because drive A is the default drive, and you did not specify another drive in the DIR command.

APPENDIX B DISK ERRORS

If a disk or device error occurs at any time during a command or program, TeleDOS returns an error message in the following format:

```
<yyy> ERROR WHILE <I/O action> ON DRIVE <d>
Abort,Ignore,Retry:_
```

In this message, <yyy> may be one of the following:

WRITE PROTECT
BAD UNIT
NOT READY
BAD COMMAND
DATA
BAD CALL FORMAT
SEEK
NON-DOS DISK
SECTOR NOT FOUND
NO PAPER
WRITE FAULT
READ FAULT
DISK

The <I/O-action> may be either of the following:

READING
WRITING

The drive <d> indicates the drive in which the error has occurred.

TeleDOS waits for you to enter one of the following responses:

A Abort. Terminate the program requesting the disk read or write.

I Ignore. Ignore the bad sector and pretend the error did not occur.

R Retry. Repeat the operation. This response is to be used when the operator has corrected the error (such as with NOT READY or WRITE PROTECT errors).

Usually, you will want to attempt recovery by entering responses in this order:

R (to try again)
A (to terminate program and try a new disk)

One other error message might be related to faulty disk read or write:

FILE ALLOCATION TABLE BAD FOR DRIVE d

This message means that the copy in memory of one of the allocation tables has pointers to nonexistent blocks. Possibly the disk was incorrectly formatted or not formatted before use. If this error persists, the disk is currently unusable and must be formatted prior to use.

APPENDIX C ANSI ESCAPE SEQUENCES

TeleDOS provides a keyboard control device driver to allow your programs to use ANSI escape sequences. The escape sequences are used to control screen cursor positioning or reassignment of the keyboard keys.

ANSI.SYS PROGRAM

The ANSI.SYS file on your TeleDOS diskette contains the device driver needed to accept ANSI escape sequences from your programs. This driver must be made resident in memory before the program is loaded.

ANSI.SYS can be loaded into memory by placing the command:

DEVICE=ANSI.SYS

into a file called CONFIG.SYS (see Appendix D). Each time TeleDOS is loaded into memory, the file CONFIG.SYS, if it exists, is read and ANSI.SYS is loaded into memory.

NOTE! The resident size of TeleDOS is increased by the size of the ANSI.SYS program.

The CONFIG.SYS file can be created using the COPY command as follows:

```
COPY CON: CONFIG.SYS<CR>
DEVICE=ANSI.SYS<CR>
^Z<CR>
```

ANSI ESCAPE SEQUENCES

The following sections describe the ANSI escape sequences used with TeleDOS.

Notes:

1. The default value is used when no values, or a zero value is entered.
2. # represents a numeric parameter. This is a decimal number specified with ASCII digits (0-9).
3. In the control sequences listed, ESC represents the 1 byte code for ESC (1BH), not the three characters ESC.

CURSOR FUNCTIONS

The following escape sequences affect the cursor position on the screen.

CUP - Cursor Position

ESC[#;#H

HVP - Horizontal & Vertical Position

ESC[#;#f

CUP and HVP move the cursor to the position specified by the parameters. The first numeric parameter specifies the line number, and the second numeric parameter specifies the column number. The default value is 1. When no parameters are specified, the cursor is moved to the home position.

CUU - Cursor Up

ESC[#A

This sequence moves the cursor up one line without changing columns. The value of # determines the number of lines moved. The default value for # is 1. The CUU sequence is ignored if the cursor is already on the top line.

CUD - Cursor Down

ESC[#B

This sequence moves the cursor down one line without changing columns. The value of # determines the number of lines moved. The default value for # is 1. The CUD sequence is ignored if the cursor is already on the bottom line.

CUF - Cursor Forward

ESC[#C

The CUF sequence moves the cursor forward one column without changing lines. The value of # determines the number of columns moved. The default value for # is 1. The CUF sequence is ignored if the cursor is already in the far right column.

CUB - Cursor Backward

ESC[D

This escape sequence moves the cursor back one column without changing lines. The value of # determines the number of columns moved. The default value for # is 1. The CUB sequence is ignored if the cursor is already in the far left column.

DSR - Device Status Report

ESC[6n

The console driver outputs a CPR sequence (see below) on receipt of the DSR escape sequence.

CPR - Cursor Position Report (from console driver to system)

ESC[#;#R

The CPR sequence reports the current cursor position through the standard input device. The first numeric parameter specifies the current line and the second numeric parameter specifies the current column.

SCP - Save Cursor Position

ESC[s

The current cursor position is saved. This cursor position can be restored with the RCP sequence (see below).

RCP - Restore Cursor Position

ESC[u

This sequence restores the cursor position to the value it had when the console driver received the SCP sequence.

ERASING

The following escape sequences affect erase functions.

ED - Erase Display

ESC[2J

The ED sequence erases the screen, and the cursor goes to the home position.

EL - Erase Line

ESC[K

This sequence erases from the cursor to the end of the line (including the cursor position).

MODES OF OPERATION

The following escape sequences affect screen graphics.

SGR - Set Graphics Rendition

ESC[#; ...;#m

The SGR escape sequence invokes the graphic functions specified by the numeric parameter(s) described below. The graphic functions remain until the next occurrence of an SGR escape sequence.

Table C-1
Graphic Function Parameters

Parameter	Parameter Function	Remarks
0	All Attributes off	
1	Bold on	
5	Blink on	
7	Reverse Video on	
8	Concealed on	(ISO 6429 standard)
30	Black foreground	(ISO 6429 standard)
31	Red foreground	(ISO 6429 standard)
32	Green foreground	(ISO 6429 standard)
33	Yellow foreground	(ISO 6429 standard)
34	Blue foreground	(ISO 6429 standard)
35	Magenta foreground	(ISO 6429 standard)
36	Cyan foreground	(ISO 6429 standard)
37	White foreground	(ISO 6429 standard)
40	Black background	(ISO 6429 standard)
41	Red background	(ISO 6429 standard)
42	Green background	(ISO 6429 standard)
43	Yellow background	(ISO 6429 standard)
44	Blue background	(ISO 6429 standard)
45	Magenta background	(ISO 6429 standard)
46	Cyan background	(ISO 6429 standard)
47	White background	(ISO 6429 standard)

SM - Set Mode

```
      ESC[=#h
or    ESC[=h
or    ESC[=0h
or    ESC[?7h
```

The SM escape sequence changes the screen width or type according to the specified numeric parameter:

Table C-2
Screen Parameters

Parameter	Parameter Function
0	40 x 25 black and white
1	40 x 25 color
2	80 x 25 black and white
3	80 x 25 color
4	320 x 200 color
5	320 x 200 black and white
6	640 x 200 black and white
7	Wrap at end of line

RM - Reset Mode

```
      ESC[=#l  (lower-case L)
or    ESC[=l
or    ESC[=0l
or    ESC[?7l
```

Parameters for RM are the same as for SM (Set Mode), except that parameter 7 will reset the wrap at the end of line mode.

KEYBOARD REASSIGNMENT

Although not part of the ANSI 3.64-1979 or ISO 6429 standard, the following keyboard reassignments are compatible with these standards.

The control sequence is:

```
     ESC[#;#;...#p
or   ESC["string";p
or   ESC[#;"string";#;#;"string";#p
or   any other combination of strings and decimal numbers
```

The final code in the control sequence (p) is one reserved for private use by the ANSI 3.64-1979 standard.

The first ASCII code in the control sequence defines which code is being mapped. The remaining numbers define the sequence of ASCII codes generated when this key is intercepted. Note that there is one exception: if the first code in the sequence is zero (NUL), then the first and second code make up an extended ASCII redefinition. The second codes indicate the following keys, or key combinations:

Second Code	Key Combination
3	NUL (null character)
15	<Shift>/<Tab>
16 - 25	<Alt>/Q, W, E, R, T, Y, U, I, O, P
30 - 38	<ALT>/A, S, D, F, G, H, J, K, L
44 - 50	<ALT>/Z, X, C, V, B, N, M
59 - 68	function keys F1 - F10
71	<Home> (7 on the numeric keypad)
72	<Cursor Up> (8 on the numeric keypad)
73	<PgUp> (9 on the numeric keypad)
75	<Cursor Left> (4 on the numeric keypad)
77	<Cursor Right> (6 on the numeric keypad)
79	<End> (1 on the numeric keypad)
80	<Cursor Down> (2 on the numeric keypad)
81	<PgDn> (3 on the numeric keypad)
82	<Ins> (0 on the numeric keypad)
83	<Del> (. on the numeric keypad)
84 - 93	<Shift>/F1 - F10 (F11 - F20)
94 - 103	<Ctrl>/F1 - F10 (F21 - F30)
104 - 113	<Alt>/F1 - F10 (F31 - F40)
114	<Ctrl>/<PrtSc>
115	<Ctrl>/<Cursor Left>
116	<Ctrl>/<Cursor Right>
117	<Ctrl>/<End>
118	<Ctrl>/<PgDn>
119	<Ctrl>/<Home>
120 - 131	<Alt>/1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, =
132	<Ctrl>/<PgUp>

Examples:

1. Reassign the Q and q key to the A and a key (and vice versa):

```
ESC[65;81p          A becomes Q
ESC[97;113p         a becomes q
ESC[81;65p          Q becomes A
ESC[113;97p         q becomes a
```

2. Reassign the F10 key to do a DIR command followed by a carriage return:

```
ESC[0;68;"dir";13p
```

The 0;68 is the extended ASCII code for the F10 key; 13 decimal is a carriage return.

APPENDIX D HOW TO CONFIGURE YOUR SYSTEM

When TeleDOS is loaded from diskette, the root directory is searched for the TeleDOS configuration file, CONFIG.SYS. If found, the text commands within the file are used to configure the system.

CHANGING THE CONFIG.SYS FILE

If there is not a CONFIG.SYS file on the TeleDOS diskette, you can use the line editor, EDLIN, to create a file; then save it on the TeleDOS diskette in your root directory.

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

1. BUFFERS=xx

   xx is a number in the range 1 to 99 indicating the number of disk buffers TeleDOS should allocate in memory. The default value is 2, and this value remains in affect until TeleDOS is restarted with a new value in the CONFIG.SYS file.

   A disk buffer is a block of memory that TeleDOS uses to hold data being read from or written to a disk when the amount of data is not an exact multiple of the sector size. Each disk buffer increases the resident size of TeleDOS by 528 bytes.

2. FILES=xx

   The maximum value for xx is 99. This value indicates the maximum number of files that can be open at the same time. The default value is eight (8).

   When files are opened, TeleDOS constructs a file control block for the file in its own memory. The amount of memory set aside for the file control blocks is determined by the value in the FILES command. The resident size of TeleDOS is increased by 40 bytes for each additional file above the default value of eight (8).

3. DEVICE=[<d>:][<path>]<filename>[.<ext>]

   This command allows you to install a device driver located in the specified file. During startup, the device driver is loaded into memory from the specified file as an extension of TeleDOS.

TeleDOS supplies standard device drivers to support the screen, the keyboard, the printer, the serial port, and the disk drives.

4. BREAK=<{ON | OFF}>

The BREAK command determines when TeleDOS checks to see if the <Ctrl>/<Break> key sequence has ben pressed. When OFF is specified, the default state, TeleDOS checks for a <Ctrl>/<Break> only when performing a screen, keyboard, printer, or asynchronous communications operation. When ON is specified, TeleDOS checks for a <Ctrl>/<Break> every time it performs any function.

The BREAK state can be changed using the TeleDOS BREAK command.

5. SHELL=[<d>:][<path>]<filename>[.<ext>]

This command allows you to specify the name of a top-level command processor to be loaded in place of COMMAND.COM.

A possible configuration file might look like this:

```
Buffers=10
Files=10
Device=ANSI.SYS
Break=ON
```

This example sets the Buffers and Files parameters to 10. The system initialization routine will load the ANSI.SYS file, loading the device driver to accept ANSI escape codes. The BREAK state is set to ON.

APPENDIX E TeleDOS DISKETTE FILE LIST

This is a list of the files that were on the TeleDOS system diskette when it was initially released. TeleVideo reserves the right to change these files without notification to the customer. If you have any questions about the files on your diskette, contact your computer store.

File	Function
COMMAND.COM	Command processor
EDLIN.COM	Line editor
PRINT.COM	Print spooler
CHKDSK.COM	Checks disks for directory errors
RECOVER.COM	Recovers a file or disk containing bad sectors
DEBUG.COM	Debugger utility program
COMP.EXE	File comparison utility
FIND.EXE	String search utility
SORT.EXE	Sort utility
EXE2BIN.EXE	Converts .EXE files
LINK.EXE	Linker program
DISKCOPY.COM	Copies diskettes
FORMAT.COM	Formats diskettes
TREE.COM	Lists directory paths
DISKCOMP.COM	Compares two diskettes
GRAPHICS.COM	Prints a graphic display on a graphics printer
ANSI.SYS	Accepts ANSI escape codes / reassigns keyboard
MODE.COM	Sets the operation mode for peripherals
SYS.COM	Transfers system files to a diskette
MORE.COM	Displays output one screen at a time
BASIC.EXE	GWBASIC interpreter

INDEX

+ (command character - Linker), 9.9
; (command character - Linker), 9.9-9.10
? wildcard character, 3.2
* wildcard character, 3.3
<COPY1>, 6.3
<COPYALL>, 6.5
<COPYUP>, 6.4
<CR>, 1.2
<INSERT>, 6.8
<NEWLINE>, 6.7
<SKIP1>, 6.9
<SKIPUP>, 6.6
<VOID>, 6.10

Abort, B.2
ANSI escape sequences, APPENDIX C
  cursor functions, C.5
  erasing, C.6
  keyboard assignment, C.8
  modes of operation, C.7
AUTOEXEC.BAT file, 2.6, 4.6-4.7
Automatic program execution, 2.6, 4.6
AUX, 3.3

Backing up diskettes, 2.4-2.6
Backup diskette, 2.4, 3.5
BASIC, 4.7, 9.3
Batch commands, 5.4, 5.53
Batch files, 4.4
  replaceable parameters, 4.8-4.9
Batch processing, 4.4-4.9
Binary files (COMP), 8.1
BREAK command, 5.5
Buffer space (COMP), 8.1

Change directory (CHDIR) command, 3.11, 5.6
CHDIR (change directory) command, 3.11, 5.6
Check disk (CHKDSK) command, 2.8, 5.7-5.9
CHKDSK (check disk) command, 2.8, 5.7-5.9
Class (Linker), 9.4, 9.11
Class name (Linker), 9.4
CLS (clear screen) command, 5.10
Command,
  arguments, 4.3
  characters (+, ; Linker), 9.9-9.10
  format, 5.1
  options, 4.2
  parameters, 4.3
  piping, 4.11

Command processor, 2.1
Command prompts (Linker),
  libraries, 9.7, 9.11
  list file, 9.7, 9.11
  object modules, 9.7, 9.10
  run file, 9.7, 9.11
COMMAND.COM file, 2.1
Commands, TeleDOS, 5.2-5.3
  BREAK, 5.5
  CD (change directory), 5.6
  CHDIR (change directory), 5.6
  CHKDSK (check disk), 5.7-5.9
  CLS (clear screen), 5.10
  COPY, 5.11-5.13
  CTTY (change TTY), 5.14
  DATE, 5.15
  DEL (delete), 5.16
  DIR (directory), 5.17
  DISKCOMP (disk compare), 5.18
  DISKCOPY (disk copy), 5.19-5.20
  ECHO, 5.53
  EXE2BIN, 5.21-5.23
  EXIT, 5.24
  External, 4.1-4.2
  FIND, 5.25-5.26
  FOR, 5.54
  FORMAT, 5.27
  GOTO, 5.55
  GRAPHICS, 5.28
  IF, 5.56
  Internal, 4.1
  MKDIR (make directory), 5.29
  MODE, 5.30-5.32
  MORE, 5.33
  PATH, 5.34-5.35
  PAUSE, 5.57
  PRINT, 5.36-5.38
  PROMPT, 5.39
  RECOVER, 5.40
  REM (remark), 5.58
  REN (rename), 5.41
  RMDIR (remove directory), 5.42
  SET, 5.43
  SHIFT, 5.59
  SORT, 5.45
  SYS (system), 5.46
  TIME, 5.47
  TREE, 5.48-5.49
  TYPE, 5.50
  VER (version), 5.51
  VERIFY, 5.51
  VOL (volume), 5.52

COMP (file comparison utility), Chapter 8
  difference reporting, 8.4
  directing output, 8.4-8.8
  error messages, 8.8
  parameters, 8.2-8.3
Compiler, 9.1, 9.2
CON, 3.3
Concatenation, 5.12-5.13
Control-C, 9-10
Control-Z, 7.3, 7.4, 7.21
COPY command, 3.4, 5.11-5.13
Copy files, 3.4
Creating directories, 3.12
CTTY (change TTY) command, 5.14
Current date, 2.1, 5.15
Current directory, 3.6
Current time, 2.2, 5.47
Cursor functions, C.5

Data error, B.2
DATE command, 5.15
Date prompt, 2.1
DEBUG commands, 10.5-10.31
  assemble, 10.7-10.8
  compare, 10.9
  dump, 10.10
  enter, 10.11-10.12
  fill, 10.13
  go, 10.14-10.15
  hex, 10.16
  input, 10.17
  load, 10.18-10.19
  move, 10.20
  name, 10.21-20.22
  output, 10.23
  quit, 10.23
  register, 10.24-10.26
  search, 10.27
  trace, 10.28
  unassemble, 10.29
  write, 10.30-10.31
DEBUG program,
  command summary, 10.6
  error messages, 10.32
  how to start, 10.1-10.2
  overview, 10.1
  parameters, 10.3-10.5
Default drive, 2.2
DEL (delete) command, 5.16
Delimiters, 4.3
Difference reporting (COMP), 8.4
DIR (directory) command, 2.7, 4.10, 5.17
Directory command (DIR), 2.7, 4.10, 5.17
Directories, hierarchial, 3.6-3.7

Directory,
  changing, 3.11-3.12
  creating, 3.12
  definition, 2.7, 3.5
  making, 3.12
  parent, 3.9, 3.11
  removing, 3.12
  working, 3.8-3.9, 3.11-3.12
Disk errors,
  abort, B.2
  data error, B.2
  file allocation table, B.3
  ignore, B.2
  not ready error, B.2
  retry, B.2
  sector not found, B.2
  seek error, B.2
  write fault error, B.2
  write protect error, B.2
DISKCOMP (disk compare) command, 5.18
DISKCOPY (disk copy) command, 2.4-2.6, 5.19-5.20
Disks, backup, 2.4, 3.5
Drive designation, 2.3, 3.1, 4.2, 8.2
Dummy parameters, batch files, 4.8-4.9

ECHO batch command, 5.53
Editing commands, 6.2
  copy characters, 6.3
  copy template, 6.5
  enter insert mode, 6.8
  new template, 6.7
  quit, 6.10
  skip multiple characters, 6.6
  skip one character, 6.9
EDLIN, 7.1-7.5
EDLIN commands,
  append lines, 7.6
  copy lines, 7.7-7.8
  delete lines, 7.9-7.10
  edit lines, 7.11
  end editing, 7.12
  insert text, 7.13-7.15
  list text, 7.16-7.17
  move, 7.18
  page, 7.19
  quit, 7.20
  replace text, 7.21-7.23
  search text, 7.24-7.26
  transfer text, 7.27
  write lines, 7.28
EDLIN errors messages, 7.29-7.31
Erasing, C.6

Error messages,
  DEBUG, 10.32
  COMP (file comparison), 8.8
  EDLIN, 7.29-7.31
  Linker, 9.16-9.18
EXE2BIN command, 5.21-5.23
EXIT command, 5.24
Extension, filename, 3.1, 4.2, 8.2
External commands, 4.1-4.2
External reference (Linker), 9.1

FAT (file allocation table), 2.7
File allocation table (FAT), 2.7
File comparison utility, Chapter 8
  difference reporting, 8.4
  directing output, 8.4-8.8
  error messages, 8.8
  parameters, 8.2-8.3
File specification, 3.2, 8.2
File system, 2.7
Filenames, 3.1, 5.1
Filename extension, 3.1, 4.2, 8.2
  .COM, 4.2
  .EXE, 4.2
Filename extensions, Linker default,
  .EXE, 9.5
  .LIB, 9.5
  .MAP, 9.5
  .OBJ, 9.5
Files,
  binary (COMP), 8.1-8.2
  comparing, 8.1-8.7
  description, 2.6
  naming, 3.1-3.2
  protecting, 3.5
  source (COMP), 8.1
  used by the Linker, 9.4-9.6
Filespec, 4.2, 5.1
Filters, 4.11
FIND command, 5.25-5.26
FOR batch command, 5.54
FORMAT command, 2.3, 5.27
Formatting diskettes, 2.3

GOTO batch command, 5.55
GRAPHICS command, 5.28
Groups (Linker), 9.4

Hidden files, 2.8, 5.46
Hierarchical directory, 3.6-3.7

IF batch commmand, 5.56
Ignore, B.2
Illegal filenames, 3.3
Input, redirection of, 4.10
Internal commands, 4.1

Keyboard reassignment, C.8

Linker,
  command characters, 9.9-9.10
  command prompts, 9.10-9.12
  definitions, 9.3-9.4
  error messages, 9.16-9.18
  overview, 9.1-9.2
  parameters, 9.12-9.14
Loading TeleDOS, 2.1
LST, 3.4

MKDIR (make directory) command, 3.12, 5.29
MODE command, 5.30-5.32
Modes of operation, C.7
MORE command, 5.33

Not ready error, B.2
NUL, 3.4

Object modules, 9.1, 9.7, 9.10
Operating system, description, 1.1
Options, command, 4.2-4.3
Ouput, redirection of, 4.10-4.11

Parameters,
  command, 4.2
  COMP (file comparison), 8.2-8.3
  Linker, 9.12-9.14
Parent directory, 3.9, 3.11
Path, 3.9-3.10, 5.1
PATH command, 3.9, 5.34-5.35
pathing, 3.9-3.10
Pathname, 3.8-3.9, 5.1
PAUSE batch command, 5.57
Pipes, 4.11
Piping, 4.11-4.12
PRINT command, 5.36-5.38
PRN, 3.4
PROMPT command, 5.39
Prompts,
  date, 2.1
  Linker, 9.7
  system, 2.2-2.3
  time, 2.2
Protecting files, 3.5

RECOVER command, 5.40
Redirecting input, 4.10
Redirecting output, 4.10
Relative address (COMP), 8.2
Relocatable load module (Linker), 9.1
REM (remark) command, 5.58
REN (rename) command, 5.41
Replaceable parameters, 4.8-4.9
Reserved filenames, 3.3-3.4
Response file (Linker), 9.6, 9.8-9.9
Retry, B.2
RMDIR (remove directory) command, 3.12, 5.42
Root directory, 3.6-3.8

Sector not found error, B.2
Seek error, B.2
Segment name, 9.4
Segments, 9.3-9.4
Separators, 2.1
SET command, 5.43
SHIFT batch command, 5.59
Shorthand notation, 3.9
SORT command, 4.11, 5.45
Source code, 9.1
Source drives, 4.4
Special characters, 3.2
Starting TeleDOS, 2.1-2.3
Subdirectory, 3.6-3.7
Syntax notation, 1.2
SYS (system) command, 5.46
System prompt, 2.2-2.3

Target drives, 4.4
TeleDOS,
  commands, (see commands, TeleDOS)
  description, 1.1
TIME command, 5.47
Time prompt, 2.2
TREE command, 5.48-5.49
Tree structure, 3.6-3.7
TYPE command, 5.50

VER (version) command, 5.51
VERIFY command, 5.51
VM.TMP file (Linker), 9.5-9.6
VOL (volume) command, 5.52
Wildcard characters, 3.2-3.3
Working directory, 3.8-3.9, 3.11-3.12
  displaying, 3.11
Write fault error, B.2
Write protect error, B.2

TeleVideo Systems, Inc.

125568-00 Rev. A B.R.

1170 Morse Avenue P.O. Box 3568 Sunnyvale, CA 94088