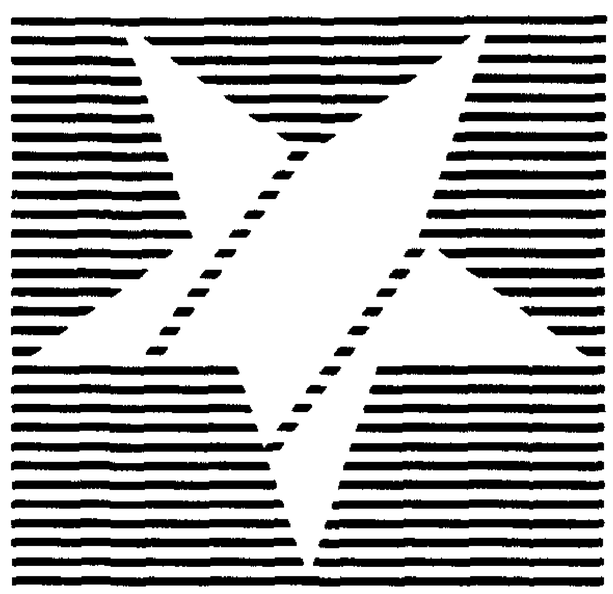
marcus bennett.

NorthStar

DOS Manual

for DIMENSION® DOS 1.2.0



03589D

North Star DIMENSION™ and DIMENSION® are trademarks or registered trademarks of North Star Computers, Inc.
DEBUG™, EDLIN™ and MS™-DOS are trademarks of Microsoft, Inc.
Microsoft® is a registered trademark of Microsoft Corporation
IBM® is a registered trademark of
International Business Machines Corporation

Copyright© 1984 by North Star Computers, Inc.
All Rights Reserved

To reorder this manual, order part number 03884

TABLE OF CONTENTS

	MANUAL CONVENTIONS	vii
	PREFACE	viii
CHAPTER 1	**ABOUT DOS FILES**	1-1
	DRIVES	1-2
	THE DOS PROMPT AND THE DEFAULT DRIVE	1-3
	WHAT IS A FILE?	1-4
	HOW DOS KEEPS TRACK OF YOUR FILES	1-5
	THE DIR COMMAND	1-6
	HOW TO NAME YOUR FILES	1-7
	WILD CARDS	1-9
	RESERVED FILENAMES	1-11
	HOW TO COPY YOUR FILES	1-12
	PROTECTING YOUR FILES	1-13
CHAPTER 2	**ABOUT DIRECTORIES**	2-1
	DIRECTORIES	2-2
	PATHS	2-6
	FILENAMES AND PATHS	2-11
	Displaying Your Working Directory	2-11
	Creating a Directory	2-12
CHAPTER 3	**LEARNING ABOUT COMMANDS**	3-1
	WHAT COMMANDS CAN DO	3-2
	TYPES OF DOS COMMANDS	3-3
	COMMAND OPTIONS	3-5
	INFORMATION COMMON TO ALL DOS COMMANDS	3-7
	BATCH PROCESSING	3-9
	The AUTOEXEC.BAT FILE	3-11
	THE AUTOEXEC.BAT FILE	3-12
	How to Create An AUTOEXEC BAT File	3-12
	Replaceable Parameters	3-14
	Executing a .BAT File With Parameters	3-16

TABLE OF CONTENTS
(cont.)

	INPUT AND OUTPUT	3-17
	Redirecting Your Output	3-18
	Filters	3-19
	Command Piping	3-20
	BACKUP AND RESTORE	3-22
	Recommended Backup Procedure	3-22
CHAPTER 4	**DOS COMMANDS**	4-1
	COMMAND FORMAT	4-2
	DOS COMMANDS	4-3
	ASSIGN Command	4-8
	BACKUP (Fixed Disk) Command	4-11
	BREAK Command	4-15
	CHDIR (Change Directory) Command	4-16
	CHKDSK (Check Disk) Command	4-17
	CLS Command	4-21
	COMMAND Command	4-22
	COPY Command	4-23
	CTTY Command	4-27
	CURSOR Command	4-28
	DATE Command	4-29
	DEL (Delete) Command	4-30
	DIR (Directory) Command	4-31
	DISKBOOT Command	4-32
	DISKCOMP Command	4-33
	DISKCOPY (Copy Diskette) Command	4-34
	ENHANCE Command	4-36
	EXE2BIN Command	4-37
	EXIT Command	4-40
	FIND Command	4-41
	FORMAT Command	4-43
	LOCAL Command	4-45
	MKDIR Command	4-49
	MORE Command	4-50
	PATH Command	4-51
	PRINT Command	4-53
	PRINTER Command	4-56

PROFILES Command 4-59
PROMPT Command 4-60
RECOVER Command 4-62
RELEASE Command 4-63
REM (Remark) Command 4-64
REN (Rename) Command 4-65
REQUEST Command 4-66
RESTART Command 4-67
RESTORE (Fixed Disk) Command 4-68
RMDIR (Remove Directory) Command 4-72
SET Command 4-73
SIGNOFF Command 4-74
SORT Command 4-75
STATS Command 4-76
TIME Command 4-77
TREE (Display Directory) Command 4-78
TYPE Command 4-80
USERS Command 4-81
VER Command 4-82
VERIFY Command 4-83
VOL (Volume) Command 4-84
Batch Processing Commands 4-85
ECHO Command 4-86
FOR Command 4-87
GOTO Command 4-88
IF Command 4-89
PAUSE Command 4-90
SHIFT Command 4-91
TIMER Command 4-92

CHAPTER 5 DOS EDITING AND FUNCTION KEYS 5-1
THE COMMAND TEMPLATE 5-2
DOS EDITING KEYS 5-3
USING THE KEYS 5-4
CONTROL CHARACTER FUNCTIONS 5-7

TABLE OF CONTENTS
(cont.)

CHAPTER 6	**THE LINE EDITOR (EDLIN)**	6-1
	EDLIN FILES	6-2
	HOW TO START EDLIN	6-3
	SPECIAL EDITING KEYS	6-5
	[F1] Key	6-6
	[F2] Key	6-7
	[F3] Key	6-8
	[Del] Key	6-9
	[F4] Key	6-10
	[Esc] Key	6-11
	[Ins] Key	6-12
	[F5] Key	6-14
	USING EDLIN COMMANDS	6-15
	EDLIN COMMANDS	
	(A)ppend Command	6-19
	(C)opy Command	6-20
	(D)elete Command	6-23
	<Line> Edit Command	6-25
	(E)nd Command	6-27
	(I)nsert Command	6-28
	(L)ist Command	6-32
	(M)ove Command	6-35
	(P)age Command	6-36
	(Q)uit Command	6-37
	(R)eplace Command	6-38
	(S)earch Command	6-42
	(T)ransfer Command	6-45
	(W)rite Command	6-46
	EDLIN ERROR MESSAGES	6-47
CHAPTER 7	**FILE COMPARISON UTILITY**	7-1
	FILE COMPARISON SYNTAX	7-3
	FC SWITCHES	7-4
	DIFFERENCE REPORTING	7-6
	REDIRECTING FC OUTPUT TO A FILE	7-7
	EXAMPLES OF FC OUTPUT	7-8
	FC ERROR MESSAGES	7-12

CHAPTER 8	**THE LINKER PROGRAM (MS-LINK)**	8-1
	DEFINITIONS YOU'LL NEED TO KNOW	8-3
	FILES THAT MS-LINK USES	8-5
	HOW TO START MS-LINK	8-7
	Prompts	8-8
	Command Line	8-9
	Response File	8-10
	COMMAND CHARACTERS	8-12
	COMMAND PROMPTS	8-14
	MS-LINK SWITCHES	8-17
	SAMPLE MS-LINK SESSION	8-21
	MS-LINK ERROR MESSAGES	8-24
CHAPTER 9	**DEBUG PROGRAM**	9-1
	HOW TO START DEBUG	9-2
	COMMAND INFORMATION	9-4
	Parameters	9-6
	DEBUG COMMANDS	
	(A)ssemble Command	9-10
	(C)ompare Command	9-12
	(D)ump Command	9-13
	Enter Command	9-15
	(F)ill Command	9-17
	(G)o Command	9-18
	(H)ex Command	9-20
	(I)nput Command	9-21
	(L)oad Command	9-22
	(M)ove Command	9-24
	(N)ame Command	9-25
	(O)utput Command	9-28
	(Q)uit Command	9-29
	(R)egister Command	9-30
	Search Command	9-33
	(T)race Command	9-34
	(U)nassemble Command	9-35
	(W)rite Command	9-37
	DEBUG ERROR MESSAGES	9-39

TABLE OF CONTENTS
(cont.)

APPENDIX A USING SIMULATED SECOND DISKETTE DRIVE	A-1
APPENDIX B DISK ERRORS	B-1
APPENDIX C ANSI ESCAPE SEQUENCES	C-1
APPENDIX D HOW TO CONFIGURE YOUR SYSTEM	D-1
APPENDIX E TROUBLESHOOTING GUIDE	E-1
INDEX	Index-1

MANUAL CONVENTIONS

Symbols and Conventions

The following conventions are used throughout this manual in descriptions of typed commands:

[ ]	Square brackets indicate that the enclosed entry is optional. Square brackets also enclose the names of typed keys, such as [Retrn] or [F1]. If you are unfamiliar with the workstation keyboard, refer to the User's Guide.
< >	Angle brackets indicate data you must enter. When the angle brackets enclose lower case text, you must type in an entry defined by the text; for example, <filename>.
{ }	Braces indicate that you have a choice between two or more entries. At least one of the entries enclosed in braces must be chosen unless the entries are also enclosed in square brackets.
...	Ellipses indicate that an entry may be repeated as many times as needed or desired.
|	A vertical bar indicates an OR statement in a command. When used with a DOS filter, the bar indicates a pipe. (On the actual keyboard, the vertical bar key appears as a split (¦) bar.)
/	Slash.
\	Backslash.
CAPS	Capital letters indicate portions of statements or commands that must be entered, exactly as shown.
BOLDFACE	Entries you type in.

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

PREFACE

What Is DOS?

The North Star DIMENSION Operating System, DOS, is based on the MicroSoft MS™-DOS operating system. It contains most features of MS-DOS, as well as enhancements made especially for this computer system.

What is an Operating System

An operating system is your "silent partner" when you are using a computer. It provides the interface between the hardware and both you (the user) and the other system software. DOS is a disk operating system that enables you to create and keep track of files, run and link programs, and access peripheral devices (such as printers) that are part of the system.

About This Manual

This manual is based on the MicroSoft MS-DOS User's Guide. It describes the basic features of MS-DOS shared by DOS, as well as the enhancements made to MS-DOS for this computer system. It contains information for people with many different levels of expertise: from those who only wish to do file operations, such as copying and deleting files, to those who wish to compile their own programs.

Chapter 1 describes starting up DOS and explains about files—what they are and how to use them. Chapter 2 explains the DOS hierarchical directory structure. Chapters 3 and 4 introduce DOS commands and explain how to use them. Chapter 5 describes the editing keys you use with DOS. Read these five chapters carefully. They contain important basic information about DOS and its commands.

More advanced users will be interested in Chapter 6, which describes the line editor, EDLIN, and Chapter 7, which explains how to use the File Comparison utility, FC. This utility is helpful when you need to compare the contents of two source or binary files.

About This Manual (cont.)

If you are writing programs and want to link separately-produced object modules and create relocatable modules, Chapter 8 describes a useful utility, MS-LINK. The DEBUG Program, outlined in Chapter 9, will be of interest to advanced users.

Appendices to this manual include: general instructions for using the diskette drive and the simulated second diskette drive; a list of the DOS error messages; a list of the ANSI escape sequences for the keyboard; system configuration instructions; and, a troubleshooting chart.

CHAPTER 1

ABOUT DOS FILES

Overview

This chapter begins with an explanation of starting up DOS. It then teaches you about DOS files. Among the topics discussed are:

- ☆ The definition of a file
- ☆ How DOS keeps track of your files
- ☆ How to name your files and how to copy them

DRIVES

Definition

This system provides storage space for your files in drives. A *drive* is a separate storage area of a fixed disk. It is also a short reference for "diskette drive." Every drive is identified by a drive letter from A to P. When the system was installed, the system manager created drives for your personal use, as well as drive C, which is shared by everyone using the system, and possibly other shared drives.

Instructions in this manual which use the term "drive" refer to both fixed disk drives and the diskette drive. In the case of the diskette drive, it refers to whichever diskette you are using in the drive.

If you desire more information about drives, refer to the User's Guide for the system before continuing with this manual.

DOS Prompt

You start up DOS each time you sign on to the system. The first display you see is this:

C>

This is the DOS prompt. The DOS prompt is issued from the *command processor,* which is a file called COMMAND.COM. This file resides on drive C. COMMAND.COM contains some of the DOS commands.

Default Drive

The letter in the prompt represents the default drive. The *default drive* is the drive that DOS searches for a file or program you want to use, or where DOS stores a new file you create unless you tell DOS to use another drive.

When you sign onto the system, drive C is the default drive. You can change the default drive to another drive, if you wish. Suppose you want to change it to your drive D. Immediately following the C> prompt, you would type:

D:[Retrn]

The colon (:) following the drive letter is a special indicator for DOS. The prompt will then be:

D>

Now DOS will search drive D for the files or programs whose names you type. Any of your drives, including the diskette drive, can become the default drive.

WHAT IS A FILE?

Definition

A *file* is a collection of related information. A file on your drive or diskette 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 drive 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 drive reside in files and each file has a unique name. You refer to files by their names. A later section of this chapter tells you how to name your files.

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

Directories

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

File Allocation Table

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

These two system areas, the directories and the File Allocation Table, enable DOS to recognize and organize the files on your drives. The File Allocation Table is created on a new drive or diskette when it is formatted with the FORMAT command. One empty directory is then created, called the *root directory*. The manager creates and formats all drives; the user is responsible for formatting his or her own diskettes. (See Chapter 4 for an explanation of the FORMAT command.)

THE DIR (SHOW DIRECTORY) COMMAND

DIR D

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

DIR [Retrn]

DOS will respond with a directory listing of all the files in the current directory on drive D. The display should look similar to this:

Volume in drive D has no label
Directory of D:

LETTER	DOC	34	9-21-84	11:48a
REPORT	DOC	286	9-30-84	4:45p
STAFF	DOC	144	7-22-84	9:22a
CHART	DOC	46	10-02-84	2:38p
MEMOS	DOC	38	10-03-84	5:06p

5 File(s) 2337688 bytes free

Directory and Filename

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

DIR MYFILE.TXT[Retrn]

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

For more information on the DIR command, refer to Chapter 4, "DOS Commands."

Filename

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

NEWFILE.EXE

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

A *filename* can be from one to eight characters long. The *filename extension* can be three or fewer characters, but must be preceded by a period. You can type any filename in small or capital letters and DOS will translate these letters into uppercase characters.

Drive Letter

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

DIR D:NEWFILE.EXE[Retrn]

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

If drive D is the default drive, DOS will search only in drive D for the filename NEWFILE and so the drive letter is not necessary. A drive letter is needed only if you want to tell DOS to look on another drive to find a file.

Types of Characters

Your filenames will probably be made up of letters and digits, but other characters are allowed, too. Legal characters for filename extensions are the same as those for filenames. You cannot use spaces in either the filename or extension. Here is a complete list of useable characters.

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

File Specification

The term file specification, or *filespec*, will be used in this manual to indicate the following format:

[<drive letter>:][\ <directory>]....
[\ <directory>]<filename> [<filename extension>]

Remember that brackets indicate optional items (except when showing key names). Angle brackets (< >) mean that you supply the text for the item. Note that the drive letter and the colon (:) are not required unless you need to indicate to DOS on which drive to search for a specific file. You do not have to give your filename a filename extension.

Examples of file specifications are:

D:MYPROG.COB
D:YOURPROG.EXT
E:NEWFILE
TEXT

Introduction

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

The ? Wild Card

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

DIR TEST?RUN.EXE[Retrn]

will list 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 * Wild Card

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

DIR TEST*.EXE[Retrn]

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

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

The * Wild Card (cont.)

The wild card designation of a dot between two asterisks (*.*) refers to all files on the drive.

★ ★ **CAUTION** ★ ★

The wild card description can be very powerful when used in DOS commands. For example, the command DEL *.*[Retrn] 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 D (regardless of their filename extensions), simply type:

DIR D:NEWFILE.*[Retrn]

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

DIR E:*.TXT[Retrn]

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

Description Certain three-letter names are reserved for the names of devices. These three-letter names cannot be used as filenames or extensions. The names are:

AUX	Used when referring to input from or output to an auxiliary device (such as a printer or diskette drive).
CON	Used when referring to keyboard input or to video display output.
LST or PRN	Used when referring to the printer.
NUL	Used when you do not want to create a particular file, but an input or output filename is required.

Even if you add drive letters or filename extensions to these filenames, they remain associated with the devices listed above. For example, D:CON.XXX still refers to the video display and is not the name of a file.

HOW TO COPY YOUR FILES

Introduction

Just as with paper files, you often need more than one copy of a computer file. The COPY command allows you to copy one or more files to another drive. You can also give the copy a different name if you specify the new name in the COPY command. Copying a file to another drive this way does not erase the original copy of the file.

The COPY command can also make copies of files on the same drive. However, you cannot make a copy of a file on the same drive unless you specify a different filename for the new copy.

Format

The format of the COPY command is:

COPY <filespec > <filespec>[Retrn]

For example,

COPY D:MYFILE.TXT A:MYFILE.TXT[Retrn]

will copy the file MYFILE.TXT on drive D to a file named MYFILE.TXT on the diskette in drive A (the diskette drive). A duplicate copy of MYFILE.TXT now exists on the diskette.

Duplicate in Same Drive

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

COPY D:MYFILE.TXT D:NEWNAME.TXT[Retrn]

You now have two copies of your file on drive D—one named MYFILE.TXT and the other named NEWNAME.TXT.

You can also copy all files on a drive to another drive (i.e., make backup copies) with the COPY command. Refer to Chapter 4, "DOS Commands," for more information about this process.

Inadvertent Errors

DOS is a powerful and useful tool in processing your personal and business information. As with any information system, inadvertent errors may occur and information may be misused. If you are processing information that cannot be replaced or is confidential, you should take steps to ensure that your data and programs are protected from accidental or unauthorized use, modification or erasure. There are simple measures you can take—such as keeping your password confidential so no one can access your drives, and making backup copies of your files with the COPY command.

CHAPTER 2

ABOUT DIRECTORIES

Overview

In Chapter 1, you learned about files. This chapter explains directories, which are structures you can create within your drives, or on a diskette, to organize your files.

DIRECTORIES

Organizing Your Files

As you learned in Chapter 1, the names of your files are kept in a directory on each drive or diskette. The directory also contains information on the size of the files, their locations on the drive, and the dates that they were created and updated.

When you are working on several different projects or sharing a drive with another user, the number of files in the directory can become large and unwieldy. You may want to organize your files into convenient categories or separate them from a co-worker's.

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

Directory Structure

In a hierarchical directory structure, the root directory is the first level in the directory structure. It is the directory that is automatically created when the manager creates and prepares a drive to store files. You can create additional directories by using the MKDIR command described in Chapter 4.

These additional directories are called **subdirectories**. A subdirectory is a directory you create to further organize a group of files. In effect, all directories made from the "root" directory are subdirectories, so sometimes the terms directory and subdirectory are used interchangeably.

Directory Structure (cont.)

Like a directory, a subdirectory doesn't hold the text of the files within it. It lists the names of both files and other subdirectories. Say you have your accounting files in one directory (which is in fact a subdirectory as shown in the illustration that follows, since ACCOUNTS is a directory made from the root). You may want to divide these files into subdirectories such as Accounts Payable and Accounts Receivable. So the subdirectory ACCOUNTS would list the subdirectories Accounts Payable and Accounts Receivable.

The directory structure grows as you create new directories for groups of files. Within each new directory, files can be added, or new subdirectories can be created. Think of hierarchical structure as a genealogical tree. The root directory is the original set of parents. The root directory then has offspring, which are called directories or subdirectories. Each subdirectory or directory then becomes the parent of another generation of subdirectories. A subdirectory has only one parent directory, but a parent directory can have several subdirectories, as you can see in the illustration that follows.

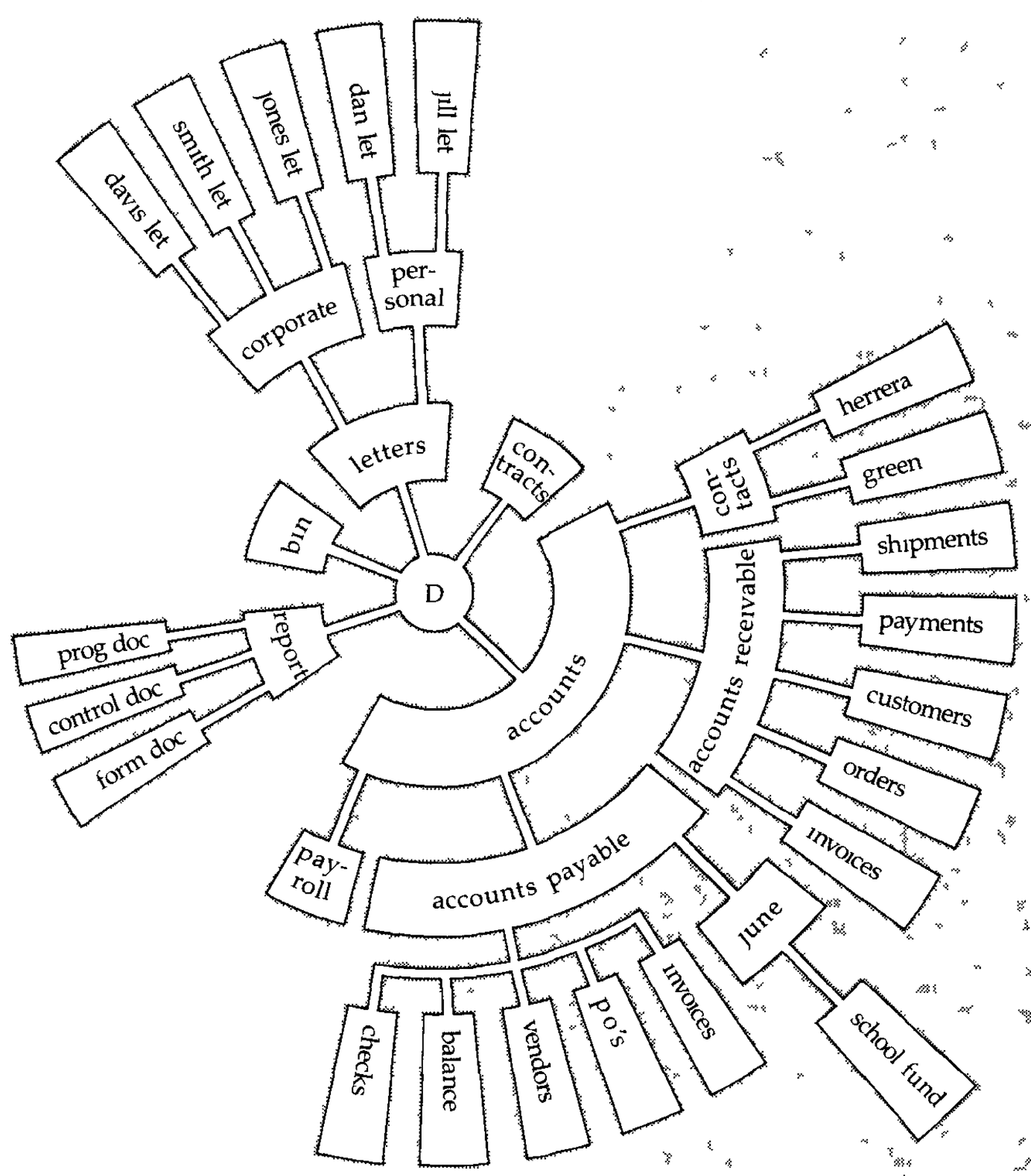
Thus, ACCOUNTS is the parent of both Accounts Receivable (AR) and Accounts Payable (AP); AR and AP are subdirectories of ACCOUNTS.

Traveling

When you first sign on the system, your current directory is the root directory. 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 the path to the desired file. Conversely, you can start where you are within the file system and travel towards the root.

Sample Directories

The following diagram illustrates a hierarchical directory structure. For clarity, descriptive names are included here. Directory and filenames can only be a maximum of eight letters. Filenames often have three-letter extensions



Sample Directories (cont.)

In the previous diagram, five subdirectories of the root directory D have been created. These are:

- ☆ A directory of all external commands, named BIN (refer to Chapter 3, "Learning About Commands," for more information on the BIN directory)
- ☆ A directory containing accounting information, named ACCOUNTS
- ☆ A directory of correspondence, called LETTERS
- ☆ A directory of reports, called REPORTS
- ☆ A directory of contracts, called CONTRACTS

The ACCOUNTS directory has subdirectories for accounts payable (AP), accounts receivable (AR), payroll (PAYROLL) files (FILES) and contracts (CONTRACTS). Within the AR and AP subdirectories are individual files such as Invoices and Orders.

The LETTERS directory has subdirectories for personal (PERSONAL) and corporate (CORPORATE) correspondence.

This organization of files and directories may not be desirable for you. But it is extremely helpful if you work on several different types of projects or share a drive with someone else.

PATHS

Introduction

If, when you tell DOS to find a file, you only give DOS the filename, DOS will search your current or working directory for that file. Your current directory is the same as your working directory. If DOS cannot find the file in the current directory, it will print the following message:

Bad command or file name

Pathnames

If the file is located elsewhere, you must tell DOS exactly where to find it by telling DOS the complete pathname. A pathname is literally the path DOS must follow in order to locate the file. The pathname may consist of several parts:

- ☆ The drive letter followed by a colon, such as "D:"
- ☆ One or more directories
- ☆ The name of the file itself

Refer to the previous illustration for the directory structure on drive D.

Finding a Subdirectory

Let's assume that your current drive is E.

If you want to see the contents of subdirectory CORPORATE, you type:

DIR D:\LETTERS\CORPORATE[Retrn]

DOS will follow the path you have specified to find your files. The steps DOS takes are the following:

1. D: tells DOS to start by going to drive D
2. \LETTERS tells DOS to find the subdirectory in the root directory
3. \ is necessary to separate LETTERS from CORPORATE.
4. CORPORATE tells DOS to look for this subdirectory within the subdirectory LETTERS.
5. Once DOS has arrived at the end of the path it will obey the DIR command and list your files.

Note: Spaces are never allowed in pathnames.

A backslash (\ ) is used in two different ways. When it immediately follows the drive letter and colon, it indicates the root directory; otherwise it is simply the symbol used to separate the names of subdirectories and filenames.

If you wanted to copy the file JONES.LET to drive E you type:

COPY D:\LETTERS\CORPORATE\JONES.LET E:

DOS follows the path (as above), locates the file and copies it to drive E.

Your Current Directory

All commands are executed from your current directory. You can find out the name of the directory you are in by issuing the DOS command CHDIR (Change Directory) For example, if your current directory is \ACCOUNTS\AP, you type:

CHDIR[Retrn]

you will see:

D:\ACCOUNTS\AP

This is your current drive and working directory (\ACCOUNTS\AP).

Change Directory

Changing from your working directory to another directory is very easy in DOS. Simply issue the CHDIR (Change Directory) command and supply a pathname. For example, suppose the working directory is \ACCOUNTS\AP. If you type:

CHDIR\ACCOUNTS[Retrn]

The working directory is changed from \ACCOUNTS\AP to \ACCOUNTS. You can travel to any directory in the directory structure by specifying its pathname.

CHDIR..[Retrn]

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

Change to Subdirectory

If you have several operations that you want to perform in the subdirectory CORPORATE, you can avoid having to type the long pathname many times. You do this by changing the current directory of D to the subdirectory CORPORATE using the CHANGE DIRECTORY command. Type:

CHDIR D:\LETTERS\CORPORATE[Retrn]

Change to Subdirectory (cont.)

Now DOS knows what part of drive D you are interested in. Once you have done this, the command lines above can be shortened. If you type:

DIR D:[Retrn]

you will get the same results as

DIR D:\LETTERS\CORPORATE[Retrn]

And if you type:

COPY D: JONES.LET E:[Retrn]

you get the same results as the longer copy command previously discussed.

Change to Root Directory

If at this point you needed to see the contents of the root directory of drive D you would type:

DIR D:\[Retrn]

Again, a backslash (\) immediately following a drive letter and colon indicates the root directory of that drive.

To change the current directory for drive D back to its root directory type:

CHDIR D:\[Retrn]

Note: Whenever you sign on, the current directory for all your drives is the root directory.

DOS Commands and Pathnames

The DOS commands that are capable of following a pathname in order to locate files are:

BACKUP	DEL	MKDIR	REMDIR
CHDIR	DIR	PATH	TYPE
COPY	FIND	RESTORE	

PATH Commands

You will find these commands explained in detail in Chapter 4. You cannot invoke a command or a program name by a pathname. The program must be in the current directory of the drive you specify. There is, however, a special way that DOS has to locate commands and programs. Please see the PATH command section of the DOS manual for details.

Shorthand Notations

DOS provides special shorthand notations for the working directory and the parent direcctory.

. DOS uses a period to indicate the name of the working directory in all hierarchical directory listings. DOS automatically creates this entry when a directory is made.

.. DOS uses two periods to indicate the name of the working directory's parent directory. If you type:

DIR ..[Retrn]

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

If you type:

DIR ..\..[Retrn]

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

Display for Subdirectory

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

```
Volume in drive D has no label
Directory of D:\ACCOUNTS\AP

.                  <DIR>          6-14-84    10:09a
..                 <DIR>          6-14-84    10:09a
JUNE               <DIR>          6-25-84    10:46a
BALANCE                35022      6-25-84     9:35a
CHECKS                 28115      6-25-84     9:30a
INVOICES               40000      6-25-84     9:45a
PURCHASE ORDERS        22345      6-25-84     9:45a
VENDORS                15542      6-25-84     9:38a
        8 File(s)                8376320 bytes free
```

Volume is a term often used for the root directory. A volume label for this drive was not assigned when the drive was formatted. Note that DOS lists both files and directories in this output. As you can see, AP has a subdirectory named JUNE. The dot (.) indicates the working directory \ACCOUNTS\AP and the double dot (..) is the shorthand notation for the parent directory \ACCOUNTS. CHECKS is a file in the \ACCOUNTS\AP directory. All of these directories and files reside on drive D.

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

Make Directory

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

MKDIR NEWDIR[Retrn]

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

Directory names follow the same rules as filenames and can have three-character extensions. However, you cannot create a subdirectory that has the same name as a file already in the parent directory. Nor can you create a file in a parent directory that has the same name as one of its subdirectories.

To put files in a new directory, use an application program to create files, or create them with the DOS line editor, EDLIN. Chapter 6, "The Line Editor (EDLIN)," describes how to use EDLIN to create and save files.

For Reference

For more information about paths and directories, see the following commands described in Chapter 4:

MKDIR RMDIR CHDIR TREE

CHAPTER 3

LEARNING ABOUT COMMANDS

Overview

Commands are a way of communicating with the computer. By typing DOS commands at your workstation, you can ask the system to perform useful tasks. This chapter explains how to use the commands.

WHAT COMMANDS CAN DO

Functions There are DOS commands that:

- ☆ Compare, copy, display, delete, and rename files
- ☆ Copy and format diskettes
- ☆ Execute system programs such as EDLIN, as well as your own programs
- ☆ Analyze and list directories
- ☆ Display the date and time
- ☆ Copy DOS system files to another drive
- ☆ Request DOS to wait for a specific period of time

Internal Commands

Of the two types of DOS commands, internal commands are the simplest and most commonly used. You cannot see these commands when you do a directory listing on your drive; they are part of the command processor, COMMAND.COM. The following internal commands are described in Chapter 4:

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

External Commands

External commands reside on drives as program files. They are read from a drive before they run. If they are not in the default drive, you must type the letter of the drive that contains them before the command name in order to execute them.

Any filename with a filename extension of .COM, .EXE or .BAT is considered an external command. For example, programs such as FORMAT.COM and USERS.EXE are external commands. Because all external commands reside on a drive (initially drive C, you can copy them into other drives later), you can create your own commands and add them to the system.

External Commands (cont)

When you type an external command, do not include its filename extension The following external commands are described in Chapter 4·

ASSIGN	DISKLOCK	PRINTER	SETDATE
BACKUP	EXE2BIN	PROFILES	SETTIME
CHKDSK	FIND	RECOVER	SIGNOFF
CURSOR	FORMAT	RELEASE	SORT
DISKBOOT	LOCAL	REQUEST	STATS
DISKCOMP	MORE	RESTART	TREE
DISKCOPY	PRINT	RESTORE	USERS

Function

Options can be included in your DOS commands to specify additional information to the system. If you do not include some options, DOS provides a default value. Refer to individual command descriptions in Chapter 4 for the default values.

To use DOS commands, type the command name, then a space and then type the option(s), if you want them.

Format Options for DOS Commands

The following are format options for all DOS commands:

Option	Description
<d:>	Refers to the drive letter.
<filename>	Refers to any valid name for a file, including an optional filename extension. The filename option does not refer to a device or to a drive letter.
<.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 letter, a filename, and an optional three letter filename extension in the following format: [<d:>]<filename>[<.ext>]
<pathname>	Refers to a pathname or filename in the following format: [<directory>\ ][<directory...>\ ]...[<filename>]

Format Options for DOS Commands (cont.)

Option	Description
switches	Options that control DOS commands. They are preceded by a forward slash (for example, /P).
arguments	Provide more information to DOS commands. You usually choose between arguments; for example, ON or OFF.

DOS Commands

General information about commands:

- ☆ Commands are usually followed by one or more options.
- ☆ Commands and options may be entered in uppercase or lowercase, or a combination of keys.
- ☆ 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 delimiters in DOS commands. In this manual, we use a space as the delimiter in commands.
- ☆ Do not separate the elements in a file specification (drive letter, filename and extension) with delimiters.
- ☆ When instructions say "Press any key," you can press any alphabet (A-Z) or numeric (0-9) key.
- ☆ In most cases you must include the filename extension when referring to a file that has one.
- ☆ You can attempt to cancel commands when they are running by holding down [Ctrl] while pressing the C key.
- ☆ Commands take effect only after you have pressed the [Retrn] key.
- ☆ Wild cards (global filename characters) and device names (for example, PRN or CON) are not allowed in the names of any commands.

DOS Commands
(cont)

- ☆ When commands produce output that moves rapidly on the video display, you can press [Ctrl] and [Num Lock], or [Ctrl] and S, to suspend the display. Press any key to resume the display. Some commands which produce a large amount of output will stop when one full screen of output has been displayed. The message "--More--" will appear at the bottom. When you're ready, press any key to continue the display.
- ☆ DOS editing and function keys can be used when entering commands. Refer to Chapter 5, "DOS Editing and Function Keys," for a complete description of these keys.
- ☆ The standard DOS prompt is the default drive letter plus a greater-than sign; for example, D>
- ☆ In the command instructions, drives are referred to as source drives and destination drives. A source drive is the drive you will be transferring information from. A destination drive is the drive you will be transferring information to.

Batch File

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

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

DOS Commands for Batch Files

Two DOS commands are available for use expressly in batch files: REM and PAUSE. REM permits you to include remarks and comments in your batch files without these remarks being executed as commands. PAUSE prompts you with the message "Strike a key when ready." If you press [Ctrl]C after the message, you can choose to either continue or cancel the batch process. REM and PAUSE are described in detail in Chapter 4.

Uses of Batch Processing

Batch processing is useful if you want to execute several DOS commands with one batch command, for example when you format and check a new diskette. To create a batch file for this purpose type:

```
REQUEST D
COPY CON NEWDISK.BAT[Retrn]
REM  This is a file to check new diskettes[Retrn]
REM  It is named NEWDISK.BAT[Retrn]
PAUSE  Insert new diskette in drive A[Retrn]
FORMAT A:[Retrn]
DIR A:[Retrn]
CHKDSK A:[Retrn]
[Ctrl]Z[Retrn]
RELEASE D
```

Uses of Batch Processing (cont.)

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

NEWDISK[Retrn]

The result is the same as if each of the lines in the batch file was typed at the workstation as individual commands The message "Strike a key when ready" will appear after the prompt to "Insert the new diskette in drive A " is displayed

General Information

The following list contains information that you should read before you create a batch file with DOS:

- ☆ Only the filename should be entered to execute the batch file. Do not enter the .BAT extension
- ☆ The commands in the file named <filename> BAT are executed after you enter the filename and press [Retrn].
- ☆ If you press [Ctrl]C while in batch processing mode, this prompt appears:

  Terminate batch job (Y/N)?

  If you press Y[Retrn], the remainder of the commands in the batch file are ignored and the DOS prompt appears.

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

  To call one batch file from another, the name of the batch file should be preceded by COMMAND /C

  EXAMPLE.BAT might consist of.

  ```
  ECHO        This shows one batch file calling another
  COMMAND /C TEST
  ECHO        Now returned to the first batch file
  ```

  where TEST.BAT is another batch file. DOS will ECHO the first line, execute TEST BAT, ECHO the third line and then complete execution of EXAMPLE.BAT.

Uses

The AUTOEXEC.BAT file is the most frequently used batch file. It allows you to automatically execute programs when you sign onto the system. An AUTOEXEC.BAT file is useful, for example, if you want to automatically start up an application, or execute a program with special commands each time you sign on. This type of file can save you the time it takes to repeatedly type commands.

AUTOEXEC.BAT files may also be used to customize DOS to your liking. You may want to include the following in your AUTOEXEC.BAT file:

ASSIGN	LOCAL	PROMPT
PATH	PRINTER	CURSOR

See Chapter 4 for explanations of these commands.

Create Your Own

Your system manager may have put an AUTOEXEC.BAT on drive C. If you have different requirements, create your own AUTOEXEC.BAT file on your drive D. DOS will note the existence of this file in drive D and will execute your AUTOEXEC and ignore any files that may exist on drive C. Your default drive will now be D.

THE AUTOEXEC.BAT FILE
How To Create An AUTOEXEC BAT File

Methods and Functions

The AUTOEXEC BAT file must be created in the root directory of your drive D You can create an AUTOEXEC.BAT file with a text editor, such as WordStar or EDLIN (Chapter 6 of this manual) or with the COPY command. This example shows how to create an AUTOEXEC BAT file that performs two operations First, it selects your current printer Then it automatically sends a directory listing of the contents of drive D to the printer.

Procedure: **How to Create An AUTOEXEC.BAT File**

1. At the D > type:

   COPY CON AUTOEXEC.BAT[Retrn]

   This statement tells DOS to copy the information you're about to type at the keyboard into the AUTOEXEC.BAT file

2. Now type:

   PRINTER <printer name>**[Retrn]**

   Substitute the name of the printer you want to use for <printer name> This statement tells DOS to invoke the PRINTER command and then set your current printer to the printer named If you type **LOCAL** for the printer name, your current printer will be the printer connected to your workstation.

3 Then type.

   DIR >PRN[Retrn]

   This statement tells DOS to run a directory of drive D and send the output to the printer

Procedure: How to Create An AUTOEXEC.BAT File

4. Type

   [Ctrl]Z[Retrn]

   to put the commands into the AUTOEXEC.BAT file.

5. Each time you sign onto the system, your current printer will be set to the printer you named. Then, a directory listing of the contents of drive D will be sent to the printer.

   Later you may want to add or delete statements in the file. You can do this with a text editor (such as EDLIN), or by deleting AUTOEXEC.BAT and then repeating the above procedure with the desired commands.

Parameters There may be times when you want to create a batch file and run it with different sets of data stored in varous DOS files.

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

Example At the D> type

COPY CON MYFILE.BAT[Retrn]
COPY %1.MAC %2.MAC[Retrn]
TYPE %2.PRN[Retrn]
TYPE %0.BAT[Retrn]
[Ctrl]Z[Retrn]

DOS responds with this message

1 File(s) copied
D>

The file MYFILE BAT, which consists of three commands, now resides on drive D

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 typed filename used to start the batch file (for example, MYFILE, or D:MYFILE).

Note: Up to ten dummy parameters (%0-%9) can be specified. Refer to the DOS command SHIFT in Chapter 4 if you wish to specify more than ten parameters.

Example (cont.)

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

Execute a Batch 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 DOS 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
```

Example

To execute the MYFILE batch process, type:

MYFILE D:PROG1 E:PROG2[Retrn]

MYFILE is substituted for %0, D:PROG1 for %1, and E:PROG2 for %2.

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

```
COPY D:PROG1.MAC E.PROG2.MAC[Retrn]
TYPE E:PROG2.PRN[Retrn]
TYPE MYFILE.BAT[Retrn]
```

INPUT AND OUTPUT

Introduction DOS always assumes that input comes from the keyboard and output goes to the video display. However, the flow of command input and output can be redirected. Input can come from a file rather than a keyboard, and output can go to a file or to a printer instead of the display. 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.

Sending Output to a File

Most commands produce output that is sent to your display. You can send this information to a file by using a greater-than sign (>) in your command. For example, the command DIR displays a directory listing of the default drive on the display. The same command can send this output to a file named MYFILES by designating the output file on the command line:

DIR >MYFILES[Retrn]

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

Appending

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

DIR >>MYFILES[Retrn]

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

Input From a File

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

SORT <NAMES >LIST1[Retrn]

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

Definition

A *filter* is a command that reads your input, transforms it in some way, and then outputs it, usually to your display 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.

Functions

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

Filter	Function
FIND	Searches for a constant string of text in a file
MORE	Displays standard output one screen at a time
SORT	Sorts text

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

Uses

You may occasionally need to have the output of one command sent as the input to another command. A sample case would be a command that produces output in columns. It could be desirable to have this columnar output sorted.

How to Pipe

To pipe, separate commands with the pipe separator, which is the vertical bar symbol (|). For example, if you type the command:

DIR | SORT[Retrn]

you will get 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.

Output to a File

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

DIR | SORT >DIREC.FIL[Retrn]

DOS will create a file named DIREC.FIL on your default drive. DIREC.FIL contains a sorted listing of the directory on the default drive, since no other drive was specified in the command.

To specify a drive other than the default drive, type:

DIR | SORT >E:DIREC.FIL[Retrn]

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

Pipeline

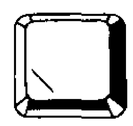
A pipeline may consist of more than two commands For example,

DIR | SORT | MORE[Retrn]

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

You will find many uses for piping commands and filters

Split Bar Key



The vertical bar key is represented as a split bar on the keyboard

BACKUP AND RESTORE
Recommended Backup Procedure

Definition

Backing up simply means making copies of the original files. If the original files become damaged in any way, you will be able to restore them with the backup files. Regularly backing up your files protects you in case of system failure, operator errors, or catastrophic data errors.

Two commands, BACKUP and RESTORE, simplify the task of regular backup. They provide a selective backup and restore function; that is, you can choose which files to backup when you type in the command. If you need to restore files at some point, you can also select those files to return to their original locations.

Alternatives

Many alternatives exist when considering the best way to handle file backup. You should study the examples shown in the commands section to determine which of the functions included in the BACKUP and RESTORE commands might prove most useful to you. We recommend the following procedure as an effective method for most situations. You can develop your own procedure as you become more familiar with the commands.

Weekly Cycle

The following procedure shows you how to backup files on a regular weekly cycle. On Monday of each work week you backup all the files from the drive you are working on. On each of the following days you backup those files that have changed since the last backup. The backup on Monday might take several diskettes to accomodate all the files on the drive. Subsequent backups, because they only copy files that have changed from the day before, will be quicker and probably require only one diskette.

Procedure: Weekly Backup Cycle

1. Each Monday backup all the files from the drive that you're using. Assuming that you're working on drive D, type:

   BACKUP D:*.* A:/S[Retrn]

   The above command means. Starting from the root directory on drive D (D:\ ), copy all files (*.*) to the diskette in drive A (A:) including all of those in subdirectories (/S).

2. Insert a new diskette each time the program prompts you to do so. Label the backup diskettes sequentially and store them in a secure location.

3. On Tuesday, perform your backup by typing:

   BACKUP D:\ *.* A:/S/M[Retrn]

   This command means: Starting from the root directory in drive D, copy all files, including those in subdirectories, to the diskette in drive A excluding those that have not been modified since the last backup (/M).

4. On Wednesday, Thursday, and Friday, repeat step #3, each time using a new diskette and labeling each diskette with the day of the week.

5. On the following Monday, return to your first set of diskettes copied on the previous Monday, and repeat step #1.

6. On each day for the rest of the week, take the diskette labeled for that day and backup all files that have been modified since the last backup (as shown in step #3).

Note: Files created by the BACKUP command cannot be used for normal processing If you want to access these files, you must use the RESTORE command to return them to the fixed disk drive. The following procedure describes the correct sequence to accomplish this

Procedure: Restoring Files to the Fixed Disk

1. Assume that due to a system problem the files that you backed up in the previous procedure were erased from the fixed disk. Take the first set of diskettes (from the full Monday backup) and insert the first in the series. To execute RESTORE, type:

   RESTORE A: D:*.*/S[Retrn]

   This commands means: From drive A (A:), copy all files (*.*) to drive D (D:) starting at the root directory (\ ) and including all subdirectories (/S).

2. As soon as RESTORE copies all the files from one diskette, you receive a prompt to insert the next diskette. Continue restoring files, inserting the diskettes in their original order, until you have restored the full set of diskettes.

3. Proceed to the backup diskette labeled Tuesday. Restore any files that have been modified since Monday's processing by typing:

   RESTORE A: D:\ *.*/S[Retrn]

   This is the same command used in the last step, but this time any files that have been updated since the day before will overwrite the less current files on the fixed disk drive.

4. Continue with the backup diskettes for the remaining days of the week using the same command as in step #3. Each diskette updates the files for that day of the week until all of the changes have been included on the fixed disk drive.

5. This completes the process to restore all files to their most recent condition.

CHAPTER 4

DOS COMMANDS

Overview

This chapter provides a detailed explanation of each DOS internal and external command, in alphabetical order.

Internal commands are the most commonly used commands. Because they are part of the command processor (COMMAND.COM, which resides on drive C) you just type them. DOS knows where to find them.

External commands are not part of the operating system, so you have to tell DOS where to find them. An external command has to be in the current drive or the path command is required to help DOS find it.

COMMAND FORMAT

How to Format DOS Commands

The following notation indicates how you should format DOS commands:

- You must enter any words shown in capital letters You can enter these keywords in any combination of upper/lowercase; DOS will convert all keywords to uppercase.
- You supply the text for any items enclosed in angle brackets < >. For example, you should enter the name of your file when <filename> is shown in the format.
- Items in square brackets [ ] are optional, except when the brackets are used to indicate a key, such as [Retrn] or [F1]. If you want to type the optional information, do not include the square brackets, only the information within the brackets.
- An ellipsis (...) indicates that you may repeat an item as many times as you want.
- You must include all punctuation where shown (with the exception of brackets), such as commas, equal signs, question marks, colons, or slashes.
- A vertical bar (|) indicates you must choose between two items.

DOS Commands

The following DOS commands are described in this chapter. Synonyms for commands are enclosed in parentheses You can type a synonym instead of the command name and achieve the same results

Note: Some of the commands, such as COPY, may be used with two diskettes. If you want to use such a command, Appendix A explains how to use two diskettes with the diskette drive (drive A) and the simulated second diskette drive (drive B)

Command	Description
ASSIGN	Reassigns drives
BACKUP	Backs up files from the fixed disk drive
BREAK	Sets [Ctrl]C check
CHDIR	Changes directories; displays working directory (CD)
CHKDSK	Scans the directory of the default or designated drive and checks for consistency
CLS	Clears screen
COMMAND	Invokes a second copy of the command processor
COPY	Copies file(s) specified
CTTY	Changes console (video display) TTY
CURSOR	Changes the size of the cursor
DATE	Displays date
DEL	Deletes file(s) specified (ERASE)
DIR	Lists requested directory entries

DOS Commands (cont)

Command	Description
DISKBOOT	Runs programs that must be executed from a diskette
DISCOMPARE	Compares any two diskettes
DISKCOPY	Copies diskettes
DISKLOCK	Protects the data on the fixed disk(s) when moving the central module
EXE2BIN	Converts executable files to binary format
EXIT	Exits COMMAND COM and returns to previous level
FIND	Searches for a constant string of text
FORMAT	Formats a diskette or drive to receive DOS files
LOCAL	Configures printers, modems or "mice" for use at a workstation
MAIL	Invokes the electronic mail utility, see the DIMENSION User's Guide.
MAINT	Invokes the system maintenance utility; see the DIMENSION System Manual
MANAGER	Places the system in or out of manager mode, see the DIMENSION System Manual
MKDIR	Makes a directory (MD)
MORE	Displays output of a command one screen at a time

DOS Commands (cont.)

Command	Description
PATH	Sets a command search path
PRINT	Places a file in the print list and displays all print lists
PRINTER	Changes the default printer and displays a list of all printer names
PROFILES	Lists the usernames and drive assignments of all users
PROMPT	Designates the DOS prompt
RECOVER	Recovers a bad diskette
RELEASE	Releases use of the diskette drive
REM	Displays a comment in a batch file
REN	Renames first file as second file (RENAME)
REQUEST	Requests use of the diskette drive
RESTART	Restarts a workstation from another workstation
RESTORE	Restores files backed up from the fixed disk drive
RMDIR	Removes a directory (RD)
SET	Sets one string value to another
SETDATE	Sets the system date, see the DIMENSION System Manual
SETTIME	Sets the system time; see the DIMENSION System Manual
SETUP	Sets up the system; see the DIMENSION System Manual

DOS Commands (cont.)

Command	Description
SIGNOFF	Signs a user off the system
SORT	Sorts data alphabetically, forward or backward
STATS	Displays statistics on space usage for fixed disks and partitions
TIME	Displays time
TREE	Displays the full directory tree
TYPE	Displays the contents of specified EDLIN file
UPDATE	Installs a new version of DOS; see the DIMENSION System Manual
USERS	Displays list of all users currently signed on
VER	Prints DOS version number
VERIFY	Verifies write operations to drives and diskettes
VOL	Prints the volume label for a drive or diskette

Batch Commands (Command Extensions)

The following DOS batch commands are also described in this chapter.

Command	Description
ECHO	Turns batch file echo feature on/off
FOR	Batch command extension
GOTO	Batch command extension
IF	Batch command extension
PAUSE	Makes a batch file pause for input during execution
SHIFT	Increases number of replaceable parameters in batch processing
TIMER	Suspends execution of a program until a specified time

Type External

Purpose Reassigns drives.

Syntax ASSIGN [<d>=<d>] .... [<d>[Retrn]

or

ASSIGN NONE[Retrn]

Comments Some applications assume that certain data or program files are stored in certain drives. By using ASSIGN, you can direct an application to look for the files in the drives of your choice.

For example, suppose an application assumes that your data files are stored in drive B when in fact they are stored in your drive D. You would type:

ASSIGN B=D[Retrn]

to tell the application to find files in drive D when it expects them to be in drive B.

Note: You must use drives that exist in your profile. For example, if you are assigned to drives C, D and E, you cannot tell ASSIGN to look for files in drive F.

If you have two or more assignments to make, you can "chain" them on the command line. For example:

ASSIGN A=C B=D C=D[Retrn]

tells the application to:

- ☆ Look in drive C for files it normally assumes are stored in drive A
- ☆ Look in drive D for files it assumes are in drive B
- ☆ Also look in drive D for files it assumes are in drive C

Note: The two assignments A=C and C=D do *not* mean that A=D.

Comments (cont.)

You may type as many assignments on the command line as will fit within the 128-character limit for a typed command line. Be sure to type a space between each one.

You can direct the program to assign two different drives to the same drive—as with B=D and C=D in the example on the previous page—but you cannot assign one drive to two drives at the same time. For example:

ASSIGN B=D B=C[Retrn]

is not allowed.

Note: Assigning B=D will not render these drives literally equal because drive B does not "follow" when you change the directory in drive D. If you want to work in a subdirectory of D you must also change the directory in B; otherwise B will point to the root directory of D rather than the subdirectory of D, and your files will not be found.

You can use an ASSIGN statement in an AUTOEXEC.BAT file if you want to set assignments automatically each time you sign on. Even if you place ASSIGN in the AUTOEXEC.BAT file, you can use ASSIGN again to make a change or an additional assignment after you sign on.

After making an assignment, ASSIGN shows you a table of the new assignments. You can also type

ASSIGN[Retrn]

to see the table of assignments before you type any new ones. ASSIGN then prompts you with:

Enter new assignments:

Type the new assignments or press **[Retrn]** for no change.

Comments (cont.)

Assignments are in effect until you sign off or cancel them with ASSIGN. There are two ways to cancel an assignment. First, suppose you have previously typed B=D and now want B to be assigned to C. You would type

ASSIGN B=C[Retrn]

to cancel B=D and set B to C.

To cancel all assignments, type:

ASSIGN NONE[Retrn]

This sets all drives back to their original assignments (A=A, B=B, etc).

★ ★ CAUTION ★ ★

If you reassign drive C to another drive, take the precaution of copying the file ASSIGN.EXE to the other drive. That way, you will be able to change the assignment back to drive C without restarting the system. You may have to copy other files from drive C as well, depending on the tasks you plan to perform.

Type External

Purpose Selectively backs up files from the fixed disk drive to diskettes.

Syntax

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

Note: Of the optional items designating the source for the backup, at least one must be specified. If no options are specified, the program responds: INVALID PARAMETERS

Comments This command performs a selective file backup based on parameters chosen by the user. Files can be selected based on a drive location, date last modified, pathname, or wild card characters. Once the file copies have been transferred to diskette by BACKUP, they can only be returned to their original locations by the RESTORE command.

BACKUP and RESTORE must work together; the files are marked in such a way that they are not accessible to other DOS utilities.

BACKUP works only with DOS-formatted diskettes. The simplest form of the format command (FORMAT A:) prepares a blank diskette for use with BACKUP. Any files contained on the inserted diskette are erased as the BACKUP begins (unless the /A parameter is chosen) When the backup diskette is filled up, the program prompts the user to insert a new diskette.

Drive C, the public drive, can only be backed up when the system is in manager mode If additional public drives are created, they also cannot be backed up unless manager mode is in effect.

Wild Card Characters

Both types of wild card characters, the asterisk (*) and the question mark (?), can be used in the filename

For example, if you type:

BACKUP C:*.REC A:[Retrn]

BACKUP will duplicate all files with the file extension .REC from the current directory of drive C onto the diskette in drive A

/S Switch

The /S switch extends the backup operation to include files in the subdirectories as well as in the specified directory. All levels of subdirectories are included by this switch, from the starting path down through the directory tree. This applies whether the pathname is specified, or whether the default value is used (the current path).

/M Switch

The /M switch backs up only those files that have been modified since the time of the last backup. This hastens the backup process by skipping over files that haven't changed

/A Switch

The /A switch appends files to the backup diskette rather than erasing the existing files. If you do not include this switch, the program prompts you to insert a diskette when you specify the BACKUP command. The /A switch can only be used with a diskette previously used as a backup diskette

/D Switch

The /D switch specifies a date; files created before the given date will not be backed up. Enter the date in the format <mm>-<dd>-<yy>. For example, if you enter:

BACKUP D: A: /D:09-11-84[Retrn]

all the files from drive D written on or after the 11th of September, 1984 will be copied over to drive A

Examples

The following examples illustrate some of the possible backup options.

To backup all files from the root directory and subdirectories of drive C onto diskette, type

BACKUP C:*.* A: /S[Retrn]

The backslash following "C." indicates the pathname, in this case the root directory. The /S switch causes all subdirectories beneath the root directory to be backed up (this includes all subdirectories within subdirectories). The wildcard characters (*.*) appear only to illustrate this example. If they were not entered, the program would default to include all files anyway.

To backup all files that have changed since the last backup and are stored in the current directory on drive C, type·

BACKUP C:*.* A:/M[Retrn]

The following example shows how to append three files taken from various subdirectories onto the same diskette.

BACKUP\DIRCTRY1\DEEP\WISDOM1.TXT A:[Retrn]
BACKUP\DIRCTRY1\DEEP\WELL\TRUE1.TXT A:/A[Retrn]
BACKUP\DIRCTRY1\KNOWLDG1.TXT A:/A[Retrn]

Since the drive is not specified for the source files, the default drive is assumed.

Operation

To use the BACKUP command, enter BACKUP followed by all necessary pathnames, wild cards and switches using the prescribed syntax The program will prompt you to insert a diskette (unless you specify the /A switch). You also receive a prompt to insert a diskette each time a backup diskette is filled up.

Operation (cont)

Filenames appear on the display as they are backed up A printed copy of the filenames can be obtained by redirecting the program's output to a printer. For example, to obtain a printed copy of your backup operation, type

BACKUP C:\ A: >PRN[Retrn]

While the backup is taking place, the filenames will be printed.

★ ★ CAUTION ★ ★

BACKUP erases all files on a diskette before performing the backup operation. Do not use this command to copy files to a diskette containing valuable data unless you specify, using the /A switch, that the files are to be added to the diskette without overwriting existing files.

Because files that are backed up by this command possess unique characteristics, they cannot be used for normal processing. Files can be returned to a form suitable for normal processing by using the RESTORE command.

Exit Codes

BACKUP uses the following exit codes. These codes can be used in conjunction with the IF batch command extension whenever you are doing batch processing.

0 Normal completion

1 No files were found to backup

3 Program was terminated by user (Control-Break)

4 Program was terminated due to error

Type Internal

Purpose Sets [Ctrl]C check.

Syntax BREAK ON | OFF

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

Type Internal

Synonym CD

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

Syntax CHDIR [<pathname>][Retrn]

Comments If your working directory is \BIN\ACCOUNTS\AP and you want to change your path to another directory (such as \BIN\ACCOUNTS\AR), type:

CHDIR \BIN\ACCOUNTS\AR[Retrn]

and DOS will change to the new directory. A shorthand notation is also available with this command:

CHDIR ..[Retrn]

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

If you want to go to your root directory, type:

CHDIR \[Retrn]

CHDIR used without a pathname displays your working directory. If your working directory is \BIN\ACCOUNTS\AP on drive D, and you type:

CHDIR[Retrn]

DOS will display·

D· \BIN\ACCOUNTS\AP

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

Type External

Purpose Scans the directory of the specified drive and checks it for consistency, gives the size of the drive, the space remaining, and the memory capacity of the workstation.

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

Comments CHKDSK should be run occasionally on each drive or diskette to check for errors in the directory. If any errors are found, CHKDSK will display error messages, if any, and then a status report. A sample status report follows:

```
5107712   bytes total disk space
   8192   bytes in 2 hidden files
  30720   bytes in bad sectors
   2048   bytes in 2 directories
 358400   bytes in 20 user files
4708352   bytes available on disk

 524288   bytes total memory
 497808   bytes free
```

You can disregard the number of bytes in bad sectors unless this number is above 208,896. DOS anticipates and compensates for this number of bytes in bad sectors. This amount is considered standard and does not cause any loss of your storage capacity.

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

Comments (cont)

You can redirect the output from CHKDSK to a file For example, type

CHKDSK D: >MYFILE[Retrn]

The errors will be sent to D:MYFILE. Do not use the /F switch if you redirect CHKDSK output

/F Switch

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

Invalid drive specification

Invalid parameter

Invalid sub-directory entry

Cannot CHDIR to file
Tree past this point not processed

First cluster number is invalid, entry truncated

Allocation error, size adjusted

Has invalid cluster, file truncated

Disk error reading FAT

Disk error writing FAT

File contains non-contiguous blocks

All specified file(s) are contiguous

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

Error	Explanation
Insufficient memory Processing cannot continue	There is not enough memory in your workstation to process CHKDSK for this drive. You must obtain more memory to run CHKDSK.
Invalid current directory Processing cannot continue	Reset the workstation and re-run CHKDSK.
Cannot CHDIR to root Processing cannot continue	The diskette or drive you are checking is bad. Try resetting your workstation. If you are using a diskette, try recovering it with RECOVER.
File is cross-linked on cluster	Make a copy of the file you want to keep, and then delete both files that are cross-linked.
X lost clusters found in y chains Convert lost chains to files (Y/N)?	If you respond Y to this prompt, CHKDSK will create a directory entry and a file for you to resolve this problem (files created by CHKDSK are named FILEnnnn.CHK). CHKDSK will then display: X bytes disk space freed If you respond N to this prompt or have not specified the /F switch, CHKDSK frees the clusters and displays: X bytes disk space would be freed

Errors (cont.)

Error	Explanation
Probable non-DOS disk Continue (Y/N)?	The diskette you are using is a non-DOS diskette. You must indicate whether or not you want CHKDSK to continue processing
Insufficient room in root directory Erase files in root and repeat CHKDSK	CHKDSK cannot process until you delete files in the root directory.
Unrecoverable error in directory Convert directory to file (Y/N)?	If you respond Y to this prompt, CHKDSK will convert the bad directory into a file. You can then fix the directory yourself or delete it.

Type Internal

Purpose Clears the video display.

Syntax CLS[Retrn]

Comments The CLS command causes DOS to send the ANSI escape sequence ESC[2J (which clears the screen) to your video display.

Type Internal

Purpose Invokes a copy of the command processor (COMMAND.COM) for specialized applications

Syntax COMMAND [<d:>][<pathname>] [/C] <string>[Retrn]

Comments Because this command deals with the operating system on an intricate level it should only be used by those thoroughly conversant in DOS. COMMAND takes a copy of COMMAND.COM from the source specified by the drive designation (<d:>) and pathname (if necessary) and executes this secondary copy.

Only one additional copy of COMMAND.COM may be invoked. To return to the initial command processor, the EXIT command can be used.

/C Switch The /C switch takes the string of characters that follows the /C and transfers them to the secondary copy of the command processor as if they had been typed in response to the DOS prompt. If you use this switch with a batch processing file, the command processor automatically executes the contents of the batch file. If you conclude the batch file with the EXIT command, the secondary command processor ceases execution and returns control to the primary command processor.

For example, if you type:

COMMAND /C BATCH1[Retrn]

the secondary command processor is invoked and the contents of the batch file, BATCH1, are executed one command at a time. If the batch file contains an EXIT command, the secondary COMMAND.COM file restores control to the original COMMAND.COM file.

Type Internal

Purpose Copies one or more files to another drive If you prefer, you can give the copies different names This command can also copy files on the same drive

Syntax COPY <filespec> [<filespec>] [/V][/A][/B][Retrn]

Comments If the second filespec is not given, the copy will be on the default drive and will have the same name as the original file (first filespec). If the first filespec is on the default drive and the second filespec is not specified, the COPY will be cancelled and DOS will display the error message

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

The optional filespec may take three forms.

If the option is a drive letter (d:) only, or a directory pathname, the original file is copied with the original filename to the designated drive or directory.

If the option is a filename only, the original file is copied to a file on the default drive with the filename specified

If the option is a full filespec, the original file is copied to a file with the filename specified

It is possible to copy files from one subdirectory to a different subdirectory on another drive Please see the discussion of pathnames in Chapter 2

/V Switch

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

The VERIFY command, discussed later in this chapter, has the same purpose as the /V switch in COPY.

Advanced Features

The /A and /B switches specify how you want the source and destination files treated during the copy operation. Files can either be handled as ASCII or binary files. When an /A or /B switch has been placed on the command line, it refers to the filespec that preceded it and all remaining filespecs on the line until another /A or /B switch is encountered.

When included with the source filespec:

- The /A switch causes the file to be copied as an ASCII text file. All data is copied up to, but not including, the end-of-file character, [Ctrl]Z. Data following [Ctrl]Z is ignored.
- The /B switch causes the file to be copied as a program or binary file. The entire file is copied based on the length recorded in the directory.

When included with the destination filespec.

The /A switch causes the end-of-file character, [Ctrl]Z, to be added to the file copy, ensuring that the text file is properly concluded.

The /B switch ensures that no end-of-file character is added to the end of the binary file

Advanced Features (cont.)

If neither switch is specified, the program assumes:

Binary files are present when copying between disk drives (as if the /B switch was in effect)

ASCII files are present when copying to or from a device other than a disk drive (as if the /A switch was in effect).

Joining Files

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

For example, if you type:

COPY A.XYZ+B.COM+B:C.TXT BIGFILE.CRP[Retrn]

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

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

COPY *.LST COMBIN.PRN[Retrn]

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

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

COPY *.LST+*.REF *.PRN[Retrn]

Joining Files (cont)

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

COPY *.LST+*.REF COMBIN.PRN[Retrn]

Limitation

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

COPY *.LST ALL.LST[Retrn]

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

Summing Files

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

COPY ALL.LST+*.LST[Retrn]

This command appends all * LST files, except ALL LST itself, to ALL LST This command will not produce an error message

Type Internal

Purpose Allows you to use a different terminal than your workstation.

Syntax CTTY <device>[Retrn]

Comments You may use CTTY with a modem to operate a remote terminal or to run programs from a remote location.

This computer system currently supports one video display (the console device, called "CON") per workstation. Other devices require a matching device driver for proper operation.

Programs that incorporate graphics, such as MAIL, or that require function keys, such as MAINT, may not run under this command.

Type External

Purpose To change the size of the cursor.

Syntax CURSOR <1 to 8> <1 to 9>[Retrn]

Comments The first number of the command determines the location of the top of the cursor relative to the space that a capital letter occupies. The second number specifies the location of the bottom of the cursor.

The top of a capital letter is represented by 1 The bottom of a letter is represented by 8.

Example CURSOR 4 8[Retrn] yields a cursor one half the height of a capital letter.

CURSOR 7 8[Retrn] yields the default cursor

Specifying 9 as the bottom of the cursor gives a full-size, full-intensity cursor regardless of the number entered for top of cursor.

CURSOR 1 9[Retrn] yields the cursor described above

Type Internal

Purpose Displays the current date

Syntax DATE[Retrn]

Comments DATE will respond with a message like

Current date is MON 10-21-84

The current date cannot be set unless you are in manager mode using a different command SETDATE

Type Internal

Synonym ERASE

Purpose Deletes all files with the designated filespec.

Syntax DEL [<filespec>][Retrn]

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

Type Internal

Purpose Lists the files in a directory.

Syntax DIR [<filespec>][/P] [/W][Retrn]

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

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

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

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

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

Type External

Purpose Runs programs that do not operate under DOS and that must be executed from a diskette.

Syntax DISKBOOT[Retrn]

Comments This command should be used with caution. If it is used with an operating system not compatible with the DIMENSION, it could cause the entire system to halt, causing other users to lose data.

With this command, you can run applications which are not designed to run under DOS and which start and run from a diskette. You do not have to request the diskette drive first. DISKBOOT does it automatically.

DISKBOOT restarts the workstation using the operating system on the diskette in the diskette drive. Since DOS will no longer be running on that workstation, none of the DOS commands will be available. In order to return the workstation to DOS, you must use the RESTART command at another workstation.

The diskette drive is automatically released if you press [ESC] to cancel the command before proceeding, or when you restart from another workstation.

Type External

Purpose Compares any two DIMENSION diskettes.

Syntax DISKCOMP[Retrn]

Comments If the formats are different this will be reported and you will be prompted to compare two more diskettes or to exit.

If the diskettes are identical this will be reported and you will be prompted to compare two more diskettes or to exit.

If the diskettes are different the differences will be reported by sector and byte. Sector numbers run from 0 to number of sectors for that type of media, and byte numbers run from 0 through 511.

[Ctrl]C and [Ctrl][Num Lock] are active and can be used in the usual way during the execution of this program.

If you wish to cancel the display of the locations of the differences between two diskettes that are substantially different without exiting this program, press the space bar and the current compare will cancel. You will then be prompted to compare two more diskettes or to exit

Type External

Purpose Performs a sector-by-sector copy operation of one or more diskettes.

Syntax DISKCOPY[Retrn]

Comments This command produces an exact copy, sector-by-sector, of any diskette with a valid DOS format. The target diskette must be formatted to match the source diskette before the DISKCOPY command can be run.

The DISKCOPY process can be canceled by pressing:

[Ctrl]C

When copying is complete, the program asks: Copy Another (Y/N?) If you respond by typing **N**, the program returns to the DOS prompt. If you respond by typing **Y**, the program directs the copying process by issuing prompts to insert the source and then the target diskette at appropriate times.

Parameters DISKCOPY does not require that any parameters be specified, any such entries on the command line are ignored.

Source and Target

When DISKCOPY is invoked the program prompts you to:

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

Insert the diskette that you wish to copy, making sure that you have a formatted diskette (with the same format as the source diskette) ready for use. When the program has copied the first portion of the source diskette, it prompts·

Insert target diskette in drive A:
Press any key when ready

If you attempt to copy to a diskette with the wrong format, the program stops you by announcing that the source and target diskettes are not the same format and the copy cannot be performed.

Once the first portion of the source diskette has been copied to the target diskette successfully, the program prompts you again:

Insert the source diskette in drive A:
Press any key when ready

Remove the target diskette and return the source diskette to the drive. The program continues to prompt you in this manner until the full contents of the source diskette have been copied to the target diskette. This cycle can be repeated as many as four times depending on the amount of data contained on the source diskette and the available memory dedicated to your workstation.

Fragmentation

Diskettes that have been subjected to considerable file creation and deletion become more difficult to access as the links connecting data to files become fragmented rather than sequential. Since DISKCOPY produces an exact copy of a diskette, it also duplicates the fragmentation. To more efficiently rearrange data from a fragmented diskette, use the COPY command to create your copy from the source diskette.

Type External

Purpose Allows certain applications to run on the DIMENSION.

Syntax ENHANCE [/A] [/B] [/OFF] [Retrn]

Comments Do not use ENHANCE unless your DIMENSION Application Installation Note specifically instructs you to do so.

Type External

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

Syntax EXE2BIN <filespec>[<filespec>][Retrn]

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

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

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

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

Kinds of Conversions (cont.)

If CS.IP is specified as 0000.100H, it is assumed that the file is to be run as a .COM file with the location pointer set at 100H by the assembler statement ORG, the first 100H bytes of the file are deleted. No segment fixups are allowed, as .COM files must be segment relocatable; that is, they must assume the entry conditions explained in the Macro Assembler Manual. Once the conversion is complete, you may rename the resulting file with a .COM extension. Then the command processor will be able to load and execute the program in the same way as the .COM programs supplied on your drive.

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 will be displayed:

File cannot be converted

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

Error Messages

If EXE2BIN finds an error, one or more of the following error messages will be displayed.

Error Message	Explanation
File not found	The file is not on the drive 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.

Error Messages (cont.)

Error Message	Explanation
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.

Type Internal

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

Syntax EXIT[Retrn]

Type External

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

Syntax FIND [/V][/C][/N] <string> [<pathname>...][Retrn]

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

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

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

The command

DIR D: | FIND /V "DAT"[Retrn]

causes DOS to display all names of the files on drive D which do not contain the string DAT.

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

/N Switch The /N switch causes each line to be preceded by its relative line number in the file. (This switch cannot be used with the /C switch.)

Strings The string should be enclosed in double quotes. For example:

FIND "Fool's Paradise" BOOK1.TXT BOOK2.TXT[Retrn]

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

Error Messages

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

FIND: Invalid number of parameters
You did not specify a string when issuing the FIND command.

FIND: Syntax error
You typed an illegal string when issuing the FIND command

FIND: File not found <filename>
The filename you have specified does not exist or FIND cannot find it.

FIND: Read error in <filename>
An error occurred when FIND tried to read the file specified in the command.

FIND: Invalid parameter <option-name> You specified an option that does not exist.

Type External

Purpose Formats a diskette or drive to accept DOS files.

Syntax FORMAT [<d:>][/O][/V][/S]

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

★ ★ **CAUTION** ★ ★

FORMAT can be a destructive command. If you accidentally use it on one of your drives, the files and directories that drive contains will be destroyed. Only use FORMAT on diskettes; or, if you are the system manager, on newly-created drives. Note that public drives can only be formatted in manager mode.

Switches The switches are described below. They must be specified in the order in which they appear above, under "Syntax."

/O Switch The /O switch causes FORMAT to produce an IBM Personal Computer DOS version 1.X compatible diskette. The /O switch causes FORMAT to reconfigure the directory with an OE5 hex byte at the start of each entry so that the diskette may be used with 1 X versions of IBM PC DOS, as well as MS-DOS 1.25/2.00 and IBM PC DOS 2.00 This switch should only be given when needed because it takes a fair amount of time for FORMAT to perform the conversion, and it noticably decreases 1.25 and 2.00 performance on diskettes with few directory entries.

/V Switch The /V switch causes FORMAT to prompt for a volume label after the drive is formatted. This is a one-to-eleven character name you can assign to the drive. The /V switch cannot be used with the /8 switch.

/1 Switch The /1 switch produces a target diskette formatted for single-sided use only. This feature allows you to format a diskette for a system with a single-sided diskette drive.

/8 Switch The /8 switch produces a target diskette formatted for use at 8 sectors per track. Although the diskette is still physically marked with 9 sectors per track, the /8 switch instructs DOS to use only 8 sectors per track. If this switch is not specified, the FORMAT command defaults to 9 sectors per track.

Type External

Purpose Configures a printer, modem and/or electronic mouse for use at a workstation.

Syntax See below.

Comments Before you can use a printer, modem or mouse connected to your workstation, you must supply some "configuration" information about it with the LOCAL command.

The five items of information needed by LOCAL are explained briefly in this section. If there is anything you don't understand, consult the system manager and the printer, modem or mouse manual before you begin.

Each item has a default value, which is the value assumed by LOCAL if you don't enter a new one.

Chart

Item	Explanation
Sending Baud Rate	The rate at which data should be *sent* to the printer or modem (or other receiving device) from any program you'll use it with. If you only have a mouse, you don't need this value. The choices are: 110 1200 150 2400 300 4800 600 9600 The default value is 2400 baud.

Chart
(cont)

Item	Explanation
Receiving Baud Rate	The rate at which data will be *received* by a program from the modem or mouse (or other sending device) you're using it with. If you only have a printer, you don't need this value The choices and the default value are the same as for the sending baud rate
Parity	Identifies the method of checking the integrity of the data that is sent or received The choices are "odd," "even" or "none," and the default value is "none "
Data Word Length	Data is typically transmitted between the device and the workstation in chunks, or "data words," which are either 7 bits or 8 bits in length. (The *bit* is the unit of measurement just below the byte.) The default length is 8 bits.
Stop Bits	Stop bits relate to the synchronization of data being sent and received. The choice is 2, if the sending baud rate is 110, or 1, if the sending baud rate is *not* 110 The default value is 1.

Rules

You can enter only one value for each item. Therefore, these rules apply.

- If you have a printer and a modem, they must have the same sending baud rate, parity, data word length and stop bits.
- If you have a printer and a mouse, they must have the same parity, data word length and stop bits.

Rules (cont.)

- If you have a modem and a mouse, they must have the same receiving baud rate, parity, data word length and stop bits.

If the two devices don't have the same values, you can't use them simultaneously. You must configure with LOCAL for one device, then reconfigure when you want to use the other.

Ways to Use Local

There are two ways to use LOCAL. The first is to type:

LOCAL[Retrn]

LOCAL then prompts you for each item of information in the order shown on the previous page. It normally displays the default value for each one. You can press **[Retrn]** to accept the default value or type a different value over it. When you finish the stop bits, press **[Retrn]**.

If you invoke LOCAL again before you sign off, the values you entered the previous time will be displayed instead of the default values.

The second way to use LOCAL is to enter the values all at once on the command line without waiting for the prompts. The order is the same: sending baud rate, receiving baud rate, parity, data word length and stop bits For example:

LOCAL 1200 1200 none 8 1[Retrn]

Type a space between each value. If the sending and receiving baud rates are the same, or if you have only one device connected to the workstation, you can type only one value for the baud rate. LOCAL will then set both rates to that value. You also don't have to type the parity, data word length or stop bits if you want their default values.

If you make a mistake when using this method, LOCAL prompts you for the values again in order

Ways to Use Local (cont.)

If you change the device connected at your workstation, use LOCAL again to configure it.

The values you set with LOCAL will remain in effect only until you sign off or use LOCAL again, unless you include a LOCAL command line (see previous example) in an AUTOEXEC.BAT file. Then the values will be set automatically each time you sign on. Refer to Chapter 3 for information about writing an AUTOEXEC.BAT file.

Type Internal

Synonym MD

Purpose Makes a new directory.

Syntax MKDIR <pathname>[Retrn]

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 \ACCOUNTS[Retrn]

will create a subdirectory \ACCOUNTS in your root directory. To create a directory named PAYROLL under \ACCOUNTS type:

MKDIR \ACCOUNTS\PAYROLL[Retrn]

Error Messages If you are unable to create a subdirectory, you may have exceeded the pathname length (64 characters), or you may have attempted to give the subdirectory the name of an already existing file.

Type External

Purpose Sends output to video display one screen at a time.

Syntax |MORE

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

Pressing any key will display another screen of information. This process continues until all the information has been displayed.

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

TYPE MYFILES.COM | MORE

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

Note: MORE creates a temporary file on the current drive which it later erases. Therefore, MORE cannot be used when the current drive is a public drive such as C.

Type Internal

Purpose Sets a command path.

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

Comments This command allows you to tell DOS which drives/directories should be searched for external commands or programs after DOS searches your working directory. DOS searches the directories in the PATH command for executable programs, that is files with extensions " EXE", ".COM", or ".BAT". The default value is no path.

The PATH is an advanced feature of DOS and not all programs have been designed to take advantage of it. In general, applications will not search the PATH for files. For instance, if you are using a word processor do not expect it to automatically search the directories listed in the PATH for the files to edit. You must explicitly tell the word processor the location of the file. Also, some programs consist of several files themselves. In general, they will not search the directories in the PATH for their various components. Such programs may start to run and then print an error message when they cannot find essential files. Your system manger will be able to give you specific instructions for running these programs.

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

To tell DOS to search drive C (its root directory), type.

PATH C:\[Retrn]

DOS will now search the root directory of drive C for external commands or programs until you set another path or sign off the system

Multiple Paths

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

PATH C:\BIN\PROGRAMS;C:\BIN\GAMES;D:\[Retrn]

tells DOS to search the directories specified by the above pathnames. DOS searches the pathnames in the order specified in the PATH command.

You may want to put a PATH command into an AUTOEXEC.BAT file. See Chapter 3 for information about the AUTOEXEC.BAT file.

Type External

Purpose Adds to a list of files to be printed on your current printer, or modifies the way a file is printed. PRINT also displays the current print lists for all printers.

Syntax PRINT [<filename>][Retrn]

Comments If you type PRINT and a filename, that file will be put into the print list for your current printer. After the file is placed in the print list, PRINT will display the print lists for all the shared printers on the system, and for the local printer connected to your workstation, if any.

You can print several files at once by typing PRINT [filename] [filename] ...[Retrn] using up to 128 characters.

If you type PRINT[Retrn] with no filename, PRINT will display the print lists without affecting them.

Note: You may not use paths or directory names when specifying files to be printed.

Switches The print utility has a number of options. These options or switches can be used by typing the command syntax as described below.

/C Switch Use the /C (copy) switch to specify the number of copies you want to print for a specific job. The syntax is:

PRINT <filename> /C=<1-255>[Retrn]

The number within the arrows indicates the number of copies you want to print.

/D Switch

The /D (delete) switch changes the status of a particular file to "canceled." In other words, when that file's turn comes up, DOS ignores the file. The syntax is:

PRINT <filename> /D[Retrn]

/F Switch

The /F (form) switch allows you to assign the form you want for a particular job; it does not change the default form. The syntax is:

PRINT <filename> /F= <A-Z> [Retrn]

The letter within the arrows specifies the type of form on which you want the file printed. When the system recognizes any form assignment that is different than the one preceding it, the printer will halt to allow you to change the form, except in the case of form Z, which will always assume a "go" condition. After changing the form at the printer, use either the /G or /R switch to restart the printer. If you don't specify a form, the system assumes the default form assigned in the PRINTER command.

/H Switch

The /H (halt) switch halts printing on your current printer. The syntax is:

PRINT /H[Retrn]

Note: The /H switch stops the print process. You must use the /G switch to restart the printer.

/G Switch

The /G (go) switch resumes printing on your current printer. The syntax is:

PRINT /G[Retrn]

/R Switch

The /R (restart) switch works the same as the /G switch but it takes the current file that was halted and restarts it from the beginning of the current copy. The syntax is:

PRINT /R[Retrn]

/T Switch

The /T (terminate) switch cancels all of your files on the print list or, if signed on as the system manager, cancels all files from the print list on the current printer. The syntax is:

PRINT /T[Retrn]

Options

The switches described above only apply to shared printers.

You can use more than one switch at a time. For example, you might want to use

PRINT D:JILL.LET /C=5 /F=B[Retrn]

in order to communicate that you want to print a file named JILL.LET, with five copies, using form B.

You may use switches to change printing options on files that are already in the print list. For example, if your print list contains D:JILL.LET /C=5 /F=B, you may change the number of copies by typing

PRINT JILL.LET /C=3

If you did not enter any options, a new file would show up in the print list as D:JILL.LET with the defaults specified in the PRINTER command.

You may also use "wild cards" to specify options. For example,

PRINT *.* /C=2

will cause each of your files in the list for the current printer to print two copies, and it will queue every file in the current directory.

Type External

Purpose Changes the current printer, sets the default form, specifies tab expansions, and specifies the form feed options PRINTER defines values for the PRINT command.

Syntax PRINTER [<printer name>][Retrn]

Comments If you simply type **PRINTER[Retrn]** the screen displays your current printer, followed by a list of the names and connections of the shared printers.

You can enter the name of the desired printer or press **[Retrn]** for no change. The printer names were assigned by the system manager when the printer was installed. The connection shows where on the central module the printers are connected.

If you type PRINTER followed by a printer name, that printer becomes your current printer. PRINTER responds with.

Current printer is <printer name>

Use LOCAL as the printer name for your local printer. Refer to the LOCAL command.

Switches

The PRINTER utility has a number of options These options or switches can be used by typing the command syntax as described below

/F Switch

The /F (form) switch sets the default form for all files printed under your username Form A is the default if no other character is specified The syntax is

PRINTER /F=<A-Z>[Retrn]

The system manager uses the alpha characters A through Z to define particular printing characteristics such as different color ribbons or paper, different paper types or sizes, or different character fonts. Ask your system manager for a list of the form definitions used on your system.

As files are printed, DOS checks the form specification. It halts the printer when it recognizes a form designation that is different from the one immediately preceding, except in the case of form Z, which will always assume a "go" condition. Refer to the PRINT command.

/T Switch

The /T (tab) switch controls the number of spaces between tabs. Many applications, such as word processing, already do this for you.

If you don't specify, the system will expand the tabs to eight spaces apart.

The syntax is·

PRINTER /T=<1-255>[Retrn]

/E Switch

The /E (eject) switch allows you to tell the printer to advance the paper to the top of the next form after printing each file. The syntax is:

PRINTER /E=ON|OFF[Retrn]

/P Switch

The /P (pause) allows you to specify the number of seconds (from 2-255) that DOS will wait before assuming that the file is completely copied to the print storage area. (The default is automatically set at 5.) When the file is determined to be complete, the file is listed in the print list. The syntax is:

PRINTER /P=<2-255>[Retrn]

You can use more than one switch at a time. For example, you might want to use

PRINTER MATRIX /F=D /T=5[Retrn]

to specify that you want to print on the printer called MATRIX, that you want to use form D, and that you want the tabs set at five spaces apart.

Type External

Purpose Lists the usernames and drive assignments of all users.

Syntax PROFILES[Retrn]

Comments Here is a sample display:

```
1.  Username:   MANAGER
    Partitions: C:1   D:2   E:7
2.  Username:   USER1
    Partitions: C:1   D:3
3.  Username:   USER2
    Partitions: C:1   D:4   E:8
4.  Username:   USER3
    Partitions: C:1   D:5   E:7
5.  Username·   USER4
    Partitions: C.1   D:6   E:7
```

To stop the display temporarily, press **[Ctrl]** and **[Num Lock]**. Press any key to resume the display. If you type the optional [>PRN], the list is printed on your current printer, instead of appearing on the display.

Type Internal

Purpose Changes the DOS command prompt.

Syntax PROMPT [<prompt-text>][Retrn]

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

Special Characters 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 carriage return, line feed sequence
s	A space (leading only)
h	A backspace
e	The '' ← '' character

Examples

PROMPT [Retrn]

sets the normal DOS prompt.

PROMPT Time = t_Date = $d[Retrn]

sets a two-line prompt which prints:

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

Type External

Purpose Recovers a file or an entire diskette containing bad sectors

Syntax RECOVER <filename | d:>[Retrn]

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

RECOVER <filename>[Retrn]

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

To recover a diskette in drive A, type:

RECOVER A:[Retrn]

If there is not enough room in the root directory, RECOVER will print a message and store 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.

★ ★ WARNING ★ ★

RECOVER A: (or any other disk) will replace ALL entries in the directory with FILENNNN.NNN and is therefore EXTREMELY dangerous.

Type External

Type Releases the use of the diskette drive.

Syntax RELEASE D[Retrn]

Comments To release the use of the diskette drive assigned with REQUEST, type:

RELEASE D[Retrn]

Type Internal

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

Syntax REM [comment][Retrn]

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

Example

```
REM   This file checks new diskettes[Retrn]
REM   It is named NEWDISK.BAT[Retrn]
PAUSE   Insert new diskette in drive A[Retrn]
FORMAT A:/S[Retrn]
DIR A:[Retrn]
CHKDSK A:[Retrn]
[Ctrl]Z[Retrn]
```

Type Internal

Synonym RENAME

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

Syntax REN <filespec> <filename>[Retrn]

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

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

Examples The following command changes the names of all files with the .LST extension to similar names with the .PRN extension:

REN *.LST *.PRN[Retrn]

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

REN D:ABODE ?D?B?[Retrn]

The file remains on drive D.

An attempt to rename a filespec to a name already present in the directory will result in the error message "File not found "

Type External

Purpose Assigns use of the diskette drive, if available to the user requesting it..

Syntax REQUEST D[Retrn]

Comments To request the diskette drive, type:

REQUEST D[Retrn]

If the diskette drive is already assigned, a message appears with the username and workstation number of the person using it. After you are finished with the diskette drive, use the RELEASE command.

Type External

Purpose Restarts a workstation that has stopped operating.

Syntax RESTART <workstation number> <username>[Retrn]

or

RESTART <workstation number> NONE[Retrn]
if no one was signed on

Comments Before using RESTART, type the USERS command to see which user is signed on to which workstation.

★ ★ CAUTION ★ ★

Restarting an active workstation can cause the person using that station to lose his or her current work.

Error Messages If you enter any incorrect values, RESTART will display an error message:

<username> is not signed on to workstation #xx.
Check the user table below.
You typed an incorrect combination of username and workstation number. RESTART then displays a table of the usernames and workstations. Select the correct values from this table.

A user signed on to that workstation.
Check the user table below.
You typed NONE for the username, but user is signed on to the selected workstation. RESTART then displays a table of the usernames and workstations. Select the correct username from this table.

Note: Before restarting a workstation, make sure that the workstation's [Caps Lock] light and [Num Lock] light are both switched off so that they will give correct indications when the workstation is reset.

Type External

Purpose Restores files backed up from partitions on the fixed disk drive to their original locations.

Syntax RESTORE <d >[<pathname>][<filespec>][/S][/P][/ I][Retrn]

Comments This command restores files, previously duplicated onto diskette by the BACKUP command, to their points of origin. The drive designation containing the backup files must be stated first, followed by the optional destination drive, pathname, filename, file extension, and any chosen switches.

RESTORE only returns files to their original locations. You cannot, for instance, restore a file to drive D if it was initially created in drive E. (The /I switch overrides this.) You can, however, working from drive D, enter the command to restore all files from the backup diskette that were created on drive E back to drive E. RESTORE does not replace other utilities like COPY; it is intended to be used if you have an operator or system failure causing loss of data on your fixed disk drive. You can then easily replace that data from your most current BACKUP diskettes.

By using the ASSIGN utility to change drive designations, you can restore files to a different drive than they were backed up from.

The optional parameters provide a means to limit the number of files restored from backup to those that meet your specifications.

Wild Card Characters

RESTORE recognizes both types of wild card characters: * and ?. For example, if you type:

RESTORE A: C:*.REC[Retrn]

RESTORE will return all files with the file extension .REC from the diskette in drive A (current directory only) to drive C (if the matching files originated on drive C).

/S Switch

The /S switch extends the restore operation to include files in the subdirectories as well as in the specified directory. All levels of subdirectories (within the given directory) on the backup diskette are included by this switch.

/P Switch

The /P switch causes the restore operation to prompt you upon detecting one of two conditions:

- ☆ Whenever the program detects a difference between the backup file and the original file on the fixed disk This generally signifies the original file has been worked with since BACKUP was last run.
- ☆ Whenever a read-only file is about to be restored.

From the prompt, you have the option of restoring the file or choosing not to. This switch protects you by warning that the data in the fixed disk file may have changed since the last backup. If you suspect that this is the case, you can avoid restoring the outdated file.

/I Switch

The /I switch tells RESTORE to restore files to any of your drives, regardless of their origin. For example, a file can be restored to drive D even if it was backed up from drive C.

Examples The following examples illustrate some of the possible restore options.

To restore all the files on the backup diskette to drive C, type:

RESTORE A: C:*.*/S[Retrn]

The backslash following the "C:" indicates the pathname, in this case the root directory. The /S switch ensures that all subdirectories beneath the root directory are restored as well.

The wild card characters (*.*) are assumed by DOS to be the default even if they are not specified.

The following examples show how to restore three separate files nested within different levels of subdirectories on the same diskette.

RESTORE \DIRCTRY1\DEEP\WISDOM1.TXT[Retrn]
RESTORE \DIRCTRY1\DEEP\WELL\TRUE1.TXT[Retrn]
RESTORE \DIRCTRY1\KNOWLDG1.TXT[Retrn]

If you have created a backup diskette containing files from three different drives (drive D, drive E, and drive F, for example), you must perform RESTORE three separate times to return all files to their original locations. Assuming that you wanted all of the files restored (without limitations) the commands would be:

RESTORE A: D:\ /S[Retrn]
RESTORE A: E:\ /S[Retrn]
RESTORE A: F:\ /S[Retrn]

Drives D, E, and F would then contain all the files last backed up from them from the root directory down through all subdirectories.

Operation

To use the RESTORE command, enter RESTORE followed by the source drive for the backup files, and any of the optional pathnames, wild cards and switches, using the prescribed syntax. Upon receiving the first prompt, insert the diskette that contains the appropriate file or files to restore. RESTORE issues a prompt if the desired file is not on the diskette, insert the next diskette in the sequence if this happens.

If you have used wild card characters, as soon as RESTORE has completed matching all filenames, you will receive a prompt to insert the next diskette.

Error Level

RESTORE uses the following codes for the error level (as recognized during batch processing):

0 Normal Completion

1 No files were found to restore

3 Program was terminated by user ([Ctrl Break] or [Esc])

4 Program was terminated due to error

Type Internal

Synonym RD

Purpose Removes a directory from a hierarchical directory structure.

Syntax RMDIR <pathname> [Retrn]

Comments This command removes a directory that is empty except for the and .. shorthand symbols.

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

RMDIR \BIN\ACCOUNTS[Retrn]

The directory has been deleted from the directory structure

Type Internal

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

Syntax SET [<string=string>][Retrn]

Comments This command is meaningful only if you want to set values that will be used by programs you have written. An application program can check all values that have been set with the SET command.

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

Type External

Purpose Signs you off the system and produces the sign-on screen.

Syntax SIGNOFF[Retrn]

Comments If any shared resources, such as the diskette drive, were assigned to you at the time you sign off, they are released. If the system manager signs off when the system is in manager mode, manager mode is exited.

Note: Before signing off a workstation, check to see that the [Caps Lock] and [Num Lock] lights are both switched off so that they will give correct indications when the workstation is reset.

Type External

Purpose SORT reads input from your keyboard, sorts the data, then writes it to your video display or files.

Syntax SORT [/R] [/+n][Retrn]

Comments SORT is used to sort a file by a certain column. There are two switches which allow you to select options:

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

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

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

SORT /R <UNSORT.TXT >SORT.TXT[Retrn]

The following command will pipe the output of the directory command to the SORT filter.

DIR | SORT /+14[Retrn]

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

DIR | SORT /+14 | MORE[Retrn]

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

Type External

Purpose Displays statistics on space usage for a fixed disk and its partitions.

Syntax STATS [<fixed disk number>][>PRN][Retrn]

Comments If you type >PRN after the fixed disk number, STATS will send the output to your current printer instead of the video display.

If you just type STATS, you will get the STATS display of fixed disk #1 on your video display.

Here is a sample display:

```
*** Fixed Disk #1
    Disk Capacity:  15351 Kbytes
    Reserved:       1860 Kbytes
    Available:      5390 Kbytes
```

PARTITION	TYPE	CAPACITY	FIRST SECTOR
1	PUBLIC	1500 Kbytes	3720
2	PERSONAL	1000 Kbytes	6720
3	PERSONAL	800 Kbytes	8720
4	PERSONAL	800 Kbytes	10320
5	SHARED	5000 Kbytes	11920
6	PERSONAL	1001 Kbytes	13920
7	SHARED	1000 Kbytes	15922
8	PERSONAL	1000 Kbytes	17922

"Disk Capacity" is the total size of the fixed disk. "Reserved" is the amount of space used by "system overhead" (not available for user partitions). "Available" is the space still available for future partitions. "First Sector" is used by technicians.

Type	Internal
Purpose	Displays the time in hours, minutes, seconds, and hundredths of seconds.
Syntax	TIME[Retrn]
Comments	TIME provides a convenient means for you to access the internal DOS clock. When the command is entered, the time is displayed in the format: hh:mm:ss.xx. hh represents the hours; valid numbers range between 0 and 23. mm represents the minutes; valid numbers range between 0 and 59. The time cannot be set unless you are in manager mode using a different command: SETTIME.

Type External

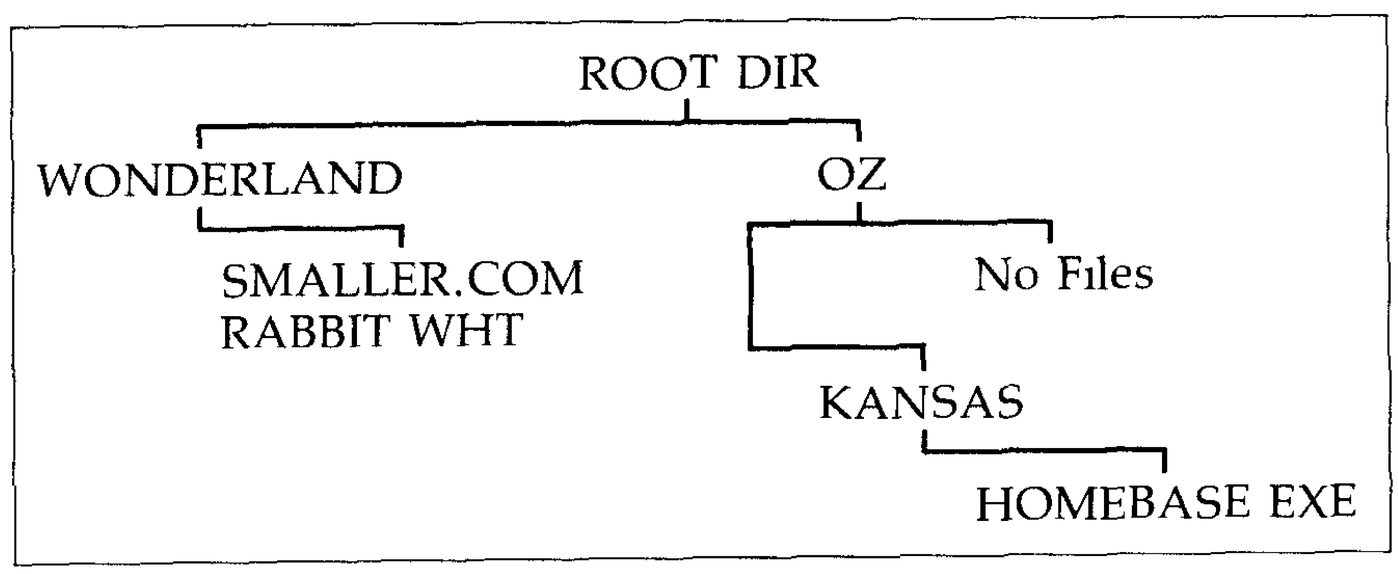
Purpose Displays the arrangement of the directory tree on a specified drive, beginning at the root directory and including all subdirectories. TREE also lists, if specified, all files contained within the subdirectories.

Syntax TREE [<d:>][/F][Retrn]

Comments This command allows you to display all paths from the root directory down through all levels of subdirectories. If a drive designation is not given, the program assumes the default drive. Regardless of whether or not you are in the root directory when you invoke TREE, the program returns to the root directory to begin the display.

/F Switch The /F switch causes all files to be displayed below their respective subdirectories.

Example Assume that the following directory structure exists on drive D under the volume label OTHERLANDS:



To display this directory arrangement, type

TREE D:/F

Example (cont.)

TREE would then display the following:

```
Directory Path Listing for Volume OTHERLANDS
Path:            \WONDERLAND
Sub-directories:None
Files:           SMALLER.COM
                 RABBIT.WHT
Path:            \OZ
Sub-directories:KANSAS
Files:           None
Path:            \OZ\KANSAS
Sub-directories:None
Files:           HOMEBASE EXE
```

I/O Redirection

The output of TREE can be redirected to a file or printer by using the redirection symbol (>) and specifying the output device or file name.

To direct the output of TREE to a file named PATHWAYS.LST, type

TREE D:/F >PATHWAYS.LST

To direct the output of TREE to a printer, type

TREE D:/F >PRN

Type Internal

Purpose Shows the contents of a file created with EDLIN

Syntax TYPE <filespec>[Retrn]

Comments Use this command to examine an EDLIN 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 [Ctrl]Z) to be sent to your computer, including bells, form feeds, and escape sequences.

Type External

Purpose Displays a list of currently signed-on users.

Syntax USERS [>PRN][Retrn]

Comments If you type >PRN after USERS, the output is sent to your current printer instead of the video display.

The listing shows the username of each user who is signed on and the number of the workstation he or she is using.

Use this command before using MANAGER to enter manager mode, or RESTART to restart a workstation.

Type Internal

Purpose Prints DOS version number.

Syntax VER[Retrn]

Comments If you want to know what version of DOS you are using, type VER. The version number will be displayed.

Type Internal

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

Syntax VERIFY [ON|OFF][Retrn]

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

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 DOS.

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

Type Internal

Purpose Displays a drive's volume label, if it exists

Syntax VOL [<d:>][Retrn]

Comments This command prints the volume label of the specified drive. If no drive is specified, DOS prints the volume label of the default drive.

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

Volume in drive x has no label

For Batch Programs

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, SHIFT and TIMER. REM, another command used in batch processing was discussed earlier in this chapter.

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

Type Internal

Purpose Turns batch echo feature on and off

Syntax ECHO [ON | OFF][<message>][Retrn]

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

If no option is specified, the current setting is displayed

Type Internal

Purpose Command extension used in batch and interactive file processing.

Syntax For batch processing:

FOR %%<c> IN (<set>) DO <command> [%%<c>][Retrn]

For interactive processing:

FOR %<c> IN (<set>) DO <command> [%<c>][Retrn]

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

<set> can be one or more items, which taken together make up the set.

If you want to execute the <command> on each of the variables, include the ending variable statement.

The %%<c> variable is 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 to each matching pattern from disk. In this case, only one such item may be in the set, and any item except for the first is ignored.

Examples FOR %%f IN (*.ASM) DO MASM %%f[Retrn]

FOR %%f IN (FOO BAR BLECH) DO REM %%f[Retrn]

The ''%%'' is needed so that after batch parameter (%0—%9) processing is done, there is one ''%'' left. If the FOR is not in a batch file, then only one ''%'' should be used.

Type Internal

Purpose Command extension used in batch file processing

Syntax GOTO <label>[Retrn]

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

:foo[Retrn]
REM looping...[Retrn]
GOTO foo[Retrn]

will produce an infinite sequence of messages

REM looping...

Starting a line in a batch file with a colon (.) 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.

Type Internal

Purpose Command extension used in batch file processing

Syntax IF <condition> <command>[Retrn]

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

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

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

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

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

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

IF NOT EXIST FOO ECHO Can't find the file[Retrn]

IF NOT ERRORLEVEL 3 LINK $1,,,;[Retrn]

Type Internal

Purpose Suspends execution of the batch file.

Syntax PAUSE [comment][Retrn]

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]C.

When the command processor encounters PAUSE, it prints the optional comment and then prints:

Strike a key when ready . . .

If you press [Ctrl]C, another prompt will be displayed:

Abort batch job (Y/N)?

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

If you want to, you can prompt the user with a message when the batch file pauses. For example, you may want the user to change diskettes in the diskette drive. In the batch file you would type:

PAUSE Please change diskettes now.[Retrn]

When the batch file runs you will see:

D>PAUSE Please change diskettes now.
Strike a key when ready . . .

Type Internal

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

Syntax SHIFT[Retrn]

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

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

then a SHIFT will result in the following:

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

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

Type External

Purpose Suspends the execution of a program until a specified time. TIMER is also used to enter manager mode. If you cannot enter manager mode because other users are signed on, the TIMER program retries manager mode every minute until all other users are signed off the system.

Syntax TIMER [hh:mm] [/M(anager)]

Comments The hh:mm represents the 24 hour system. If you type in a valid time, the program waits until that time to begin. The time you give can be any time within the next 24 hours. Seconds are disregarded.

CHAPTER 5

DOS EDITING AND FUNCTION KEYS

Overview This chapter explains:

- ☆ The editing keys
- ☆ Key combinations you can use with DOS

THE COMMAND TEMPLATE

Introduction

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

Features

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

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

How it Works

When a user types a command to DOS on the command line and presses [Retrn], the command is automatically sent to the command processor (COMMAND.COM) for execution. At the same time, a copy of this command is sent to the template You can now recall the command from the template or modify the template with DOS special editing keys.

Table of Editing Keys

The following table contains a list of the editing keys. The keys are more fully described in Chapter 6, The Line Editor (EDLIN), where they can be used to edit files.

Key	Command	Function
[F1]	COPY1	Copies one character from the template to the command line
[F2]	COPYUP	Copies characters up to the character specified in the template and puts these characters on the command line
[F3]	COPYALL	Copies all remaining characters in the template to the command line
[Del]	SKIP1	Skips over (does not copy) a character in the template
[F4]	SKIPUP	Skips over (does not copy) the characters in the template up to the character specified
[Esc]	VOID	Voids the current input; leaves the template unchanged
[Ins]	INSERT	Enters/exits insert mode
[F5]	NEWLINE	Makes the new line the new template
[F6]	[Ctrl]Z	Puts a [Ctrl]Z (1AH) end-of-file character in the new template

The [F3] Key

If you type the following command:

DIR PROG.COM[Retrn]

DOS displays information about the file PROG.COM on your display. The command line is also saved in the template. To repeat the command, just press:

[F3][Retrn]

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

[F3]DIR PROG.COM**[Retrn]**

Notice that pressing [F3] causes the contents of the template to be copied to the command line; pressing [Retrn] causes the command line to be sent to the command processor for execution.

The [F2] Key

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

[F2]C

Typing [F2] (COPYUP) and then C copies all characters from the template to the command line, up to but not including C. DOS displays:

DIR PROG.

Now type:

.ASM

The result is:

DIR PROG.ASM

The [F2] Key (cont.)

The command line DIR PROG.ASM is now in the template and ready to be sent to the command processor for execution. To do this, press:

[Retrn]

Automatic Replacement

Now assume that you want to execute the following command:

TYPE PROG.ASM

To do this, type·

TYPE[Ins] [F3][Retrn]

Notice that when you are typing, the characters are entered directly into the command line and overwrite corresponding characters in the template. This automatic replacement feature is turned off when you press the insert key. Thus,the characters ''TYPE'' replace the characters ''DIR'' in the template. To insert a space between ''TYPE'' and ''PROG.ASM'', you pressed [Ins] (INSERT) and then the space bar. Finally, to copy the rest of the template to the command line, you pressed [F3] and then [Retrn]. The command TYPE PROG.ASM has been processed by DOS, and the template becomes TYPE PROG.ASM.

More Examples

If you had misspelled TYPE as BYTE, a command error would have occurred. Still, instead of throwing away the whole command, you could save the misspelled line before you press [Retrn] by creating a new template with the [F5] (NEWLINE) key·

BYTE PROG.ASM**[F5]**

You could then edit this erroneous command by typing:

T[F1]P[F3][Retrn]

The [F1] Key

The [F1] (COPY1) key copies a single character from the template to the command line. The resulting command line is then the command that you want:

TYPE PROG ASM

As an alternative, you can use the same template containing BYTE PROG ASM and then use the [Del] (SKIP1) and [Ins] keys to achieve the same result:

[Del][Del][F1][Ins]YP[F3][Retrn]

To illustrate how the command line is affected as you type, examine the keys typed on the left; the screen displays are shown in the center and their effect on the command line is shown on the right:

Keys Typed	Screen Display	Effect
[Del]	_	Skips over 1st template character
[Del]	_	Skips over 2nd template character
[F1]	T	Copies 3rd template character
[Ins]YP	TYP	Inserts two characters
[F3]	TYPE PROG.ASM	Copies rest of template

Notice that [Del] does not affect the command line. It affects the template by deleting the first character. Similarly, [F4] (SKIPUP) deletes characters in the template, up to but not including a given character

These special editing keys can add to your effectiveness at the keyboard. The next section describes control character functions that can also help when you are typing commands.

Introduction A control character function is a function that affects the command line. You have already learned about [Ctrl]C and [Ctrl]S. Other control character functions are described below.

Remember that when you type a control character, such as [Ctrl]C, you must hold down the [Ctrl] key and then press the C key.

The following table shows the control character functions.

Control Character	Function
[Ctrl]N	Toggles echoing of output to line printer
[Ctrl]C	Cancels current command
[Ctrl]H	Removes last character from command line, and erases character from display
[Ctrl]J	Inserts physical end-of-line, but does not empty command line, use the [Ctrl]J keys to extend the current logical line beyond the physical limits of one display screen
[Ctrl]P	Toggles display output to line printer
[Ctrl]S	Suspends output on the display, press any key to resume
[Ctrl]X	Cancels the current line; empties the command line; and then outputs a back slash (\ ), carriage return, and line feed, template used by the special editing commands is not affected

CHAPTER 6

THE LINE EDITOR (EDLIN)

Overview

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.

Text Placement

The text in files created or edited by EDLIN is divided into lines, each up to 253 characters long. Line numbers are generated and displayed by EDLIN during the editing process, but are not actually present in the saved file.

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

Creating a File

To start EDLIN, type:

EDLIN <filespec>[Retrn]

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

```
New file
*
```

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

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

Editing an Existing File

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

```
End of input file
*
```

You can then edit the file using EDLIN editing commands.

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

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

Saving the File

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

Do not try to edit a file with a filename extension of .BAK because EDLIN assumes that any .BAK file is a backup file. If you find it necessary to edit such a file, rename the file with another extension (using the DOS RENAME command discussed in Chapter 4), then start EDLIN and specify the new <filespec>.

Keys to Use The special editing keys and template discussed in Chapter 5 can be used to edit your text files. These keys are discussed in detail in this section.

The table below summarizes the keys, commands, and functions. Descriptions of the special editing keys follow the table.

Key	Command	Function
[F1]	COPY1	Copies one character from the template to the | new line
[F2]	COPYUP	Copies all characters from the template to the new line, up to the character specified
[F3]	COPYALL	Copies all remaining characters in the template to the display
[Del]	SKIP1	Does not copy (skips over) a character
[F4]	SKIPUP	Does not copy (skips over) the characters in the template, up to the character specified
[Esc]	VOID	Voids the current input; leaves the template unchanged
[Ins]	INSERT	Enters/exits insert mode
[F5]	NEWLINE	Makes the new line the new template

SPECIAL EDITING KEYS
[F1] Key

Purpose Copies one character from the template to the command line.

Comments Pressing [F1] copies one character from the template to the command line. When [F1] is pressed, one character is inserted in the command line and insert mode is automatically turned off.

Example Assume that the display shows:

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

At the beginning of the editing session, the cursor is positioned at the beginning of the line. Pressing [F1] copies the first character (T) to the second of the two lines displayed:

1:*This is a sample file
[F1] 1:*T

Each time [F1] is pressed, one more character appears:

[F1] 1:*Th
[F1] 1:*Thi
[F1] 1:*This

and so forth.

Purpose Copies multiple characters up to a given character.

Comments Pressing [F2] copies all characters up to a given character from the template to the command line. The given character is the next character typed after [F2]; it is not copied or displayed on the display. Pressing [F2] causes the cursor to move to the single character that is specified in the command. If the template does not contain the specified character, nothing is copied. Pressing [F2] also automatically turns off insert mode.

Example Assume that the display shows:

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

At the beginning of the editing session, the cursor is positioned at the beginning of the line. Pressing [F2] copies all characters up to the character specified immediately after [F2].

1:*This is a sample file
[F2]p 1:*This is a sam

SPECIAL EDITING KEYS
[F3] Key

Purpose Copies template to command line.

Comments Pressing [F3] copies all remaining characters from the template to the command line. Regardless of the cursor position at the time [F3] is pressed, the rest of the line appears, and the cursor is positioned after the last character on the line.

Example Assume that the display shows:

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

At the beginning of the editing session, the cursor is positioned at the beginning of the line. Pressing [F3] copies all characters from the template (shown in the upper line displayed) to the line with the cursor (the lower line displayed):

```
     1:*This is a sample file        (template)
[F3] 1:*This is a sample file.       (command line)
```

Also, insert mode is automatically turned off.

Purpose Skips over one character in the template.

Comments Pressing [Del] skips over one character in the template. Each time you press [Del], one character is not copied from the template. The action of [Del] is similar to [F1] (COPY1), except that [Del] skips a character in the template rather than copying it to the command line.

Example Assume that the display shows:

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

At the beginning of the editing session, the cursor is positioned at the beginning of the line. Pressing [Del] skips over the first character (T).

```
      1:*This is a sample file
[Del] 1:*
```

The cursor position does not change and only the template is affected. To see how much of the line has been skipped over, press **[F3]**, which moves the cursor beyond the last character of the line.

```
      1:*This is a sample file.
[Del] 1:*
[F3]  1:*his is a sample file.
```

SPECIAL EDITING KEYS
[F4] Key

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

Comments Pressing [F4] skips over all characters up to a given character in the template. This character is not copied and is not shown on the display. If the template does not contain the specified character, nothing is skipped over. The action of [F4] is similar to [F2], except that [F4] skips over characters in the template rather than copying them to the command line.

Example Assume that the display shows:

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

At the beginning of the editing session, the cursor is positioned at the beginning of the line. Pressing [F4] skips over all the characters in the template up to the character pressed after [F4]:

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

The cursor position does not change. To see how much of the line has been skipped over, press [F3] to copy the template. This moves the cursor beyond the last character of the line:

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

Purpose Quits input and empties the command line.

Comments Pressing [Esc] empties the command line, but it leaves the template unchanged. [Esc] also prints a back slash (\ ), carriage return, and line feed, and turns insert mode off. The cursor is positioned at the beginning of the line. Pressing [F3] copies the template to the command line and the command line appears as it was before [Esc] was pressed.

Example Assume that the display shows:

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

At the beginning of the editing session, the cursor is positioned at the beginning of the line. Assume that you want to replace the line with ''Sample File:''

1:*This is a sample file.
1:***Sample File**

To cancel the line you just entered (Sample File), and to keep ''This is a sample file.'', press **[Esc]**

Notice that a back slash appears on the Sample File line to tell you it has been cancelled.

1:*This is a sample file.
[Esc] 1:*Sample File \
1:

Press **[Retrn]** to keep the original line, or to perform any other editing functions. If **[F3]** is pressed, the original template is copied to the command line. (pg 6-11)

[F3] 1: This is a sample file.

SPECIAL EDITING KEYS
[Ins] Key

Purpose Enters \ exits insert mode.

Comments Pressing [Ins] causes EDLIN to enter or exit insert mode. The current cursor position in the template is not changed.

Normally, EDLIN is in "replace" mode, in which the characters you type overstrike and replace characters in the template.

Example Assume that the screen shows:

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

Assume that you then press **[F2]m, [Ins]lary, [Ins] tax,** and then **[F3]**:

```
                1:*This is a sample file.
[F2]m           1:*This is a sa__
[Ins]lary       1:*This is a salary__
[Ins] tax       1:*This is a salary tax__
[F3]            1:*This is a salary tax file.__
```

You used [Ins] the first time to insert "lary" and then used it again to turn insert mode off in order to replace "mple" with "tax".

In insert mode, the cursor moves as each character is inserted. However, when you have finished inserting characters, the cursor will be positioned at the same character as it was before the insertion began. Thus, characters are inserted in front of the character to which the cursor points.

Here is another example of insert mode. Assume that the display shows:

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

Example (cont.)

At the beginning of the editing session, the cursor is positioned at the beginning of the line. Assume that you press **[F2] f**

```
      1:*This is a sample file
[F2]f 1:*This is a sample
```

Now press **[Ins]** and insert the characters "edit" and a space:

```
            1:*This is a sample file.
[F2]f       1:*This is a sample
[Ins]edit   1:*This is a sample edit
```

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

```
      1:*This is a sample edit
[F3]  1:*This is a sample edit file.
```

If you pressed **[Retrn]** instead of [F3], the remainder of the template would be truncated, and the command line would end at the end of the insert:

[Ins]edit[Retrn] 1:*This is a sample edit

To exit insert mode, simply press **[Ins]** again.

Purpose Creates a new template.

Comments Pressing [F5] copies the current command line to the template. The contents of the old template are deleted. Pressing [F5] outputs an ''at sign'' character (@), a carriage return and a line feed. The command line is also emptied and insert mode is turned off.

Note: [F5] performs the same function as [Esc], except that the template is changed and an @ is printed instead of a back slash (\ ).

Example Assume that the display shows:

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

At the beginning of the editing session, the cursor is positioned at the beginning of the line. Assume that you enter **[F2]m, [Ins]lary** (to turn insert mode on and insert ''lary''), **[Ins] tax** (to turn insert mode off and replace ''mple'' with ' tax'') and then **[F3]**:

	1:*This is a sample file.
[F2]m	1:*This is a sa_
[Ins]lary	1:*This is a salary_
[Ins] tax	1:*This is a salary tax_
[F3]	1:*This is a salary tax file._

At this point, assume that you want this line to be the new template, so you press **[F5]**

[F5] 1:*This is a salary tax file.@

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

General Information

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

Pathnames

Pathnames are acceptable as options to commands. For example, typing EDLIN \BIN\ACCOUNTS\AP\OFFICE will allow you to edit the OFFICE file in the subdirectory AP.

Line Numbers

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. For example:

-10,+10L[Retrn]

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

Multiple Commands

Multiple commands may be issued on one command line. When you issue a command to edit a single line using a line number (<line>), a semicolon must separate commands on the line. Otherwise, one command may follow another without any special separators. In the case of a Search or Replace command, the <string> may be ended by a [Ctrl]Z instead of a [Retrn]. For example the following command line edits line 15 and then displays lines 10 through 20 on the screen:

15;-5,+5L[Retrn]

The command line in the next example searches for "This string" and then displays 5 lines before and 5 lines after the line containing the matched string. If the search fails, then the displayed lines are those line numbers relative to the current line.

SThis string[Ctrl]Z-5,+5L[Retrn]

Space

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

Control Characters

It is possible to insert a control character (such as [Ctrl]C) into text by using the quote character ([Ctrl]V) before it while in insert mode. [Ctrl]V tells DOS to recognize the next capital letter typed as a control character. It is also possible to use a control character in any of the string arguments of Search or Replace by using the special quote character. For example:

S[Ctrl]VZ[Retrn]

will find the first occurrence of [Ctrl]Z in a file.

R[Ctrl]VZ[Ctrl]Z foo[Retrn]

will replace all occurrences of [Ctrl]Z in a file by foo.

S[Ctrl]VC[Ctrl]Z bar[Retrn]

will replace all occurrences of [Ctrl]C by bar.

It is possible to insert [Ctrl]V into the text by typing

[Ctrl]VV

End-of-File

The [Ctrl]Z character ordinarily tells EDLIN, ''This is the end of the file.'' If you have [Ctrl]Z characters elsewhere in your file, you must tell EDLIN that these other control characters do not mean end-of-file. Use the /B switch to tell EDLIN to ignore any [Ctrl]Z characters in the file and to show you the entire file.

EDLIN Command Summary

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

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

Command Options

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

<line>

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

<line> may be specified one of the following 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.

<line> (cont.)	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.
	[Retrn]	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 [Retrn] for a yes response, or for any other key for a no response.
<string>		<string> represents text to be found, to be replaced, or to replace other text. The <string> option is used only with the Search and Replace commands. Each <string> must be ended by a [Ctrl]Z or a [Retrn] (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.

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

Syntax [<n>]A[Retrn]

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

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

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

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

EDLIN COMMANDS
(C)opy 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.

Syntax [<line>],[<line>],<line>[,<count>]C[Retrn]

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

The first <line> is the beginning line, the second <line> is the ending line, and the third <line> is the destination line. The first two commas are not optional. The comma preceding <count> is optional.

The line numbers must not overlap or you will get an "Entry error" message. For example, 3,20,15C would result in an error message.

Example 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.
```

Example (cont.)

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

1,6,7C[Retrn]

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.
```

Copy and Insert

If you want to place the text within other text, the third <line> option should specify the line before which you want the copied text to appear. For example, assume that you want to copy lines and insert them within the following file:

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

Copy and Insert (cont.)

The command **3,6,9C[Retrn]** 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.
```

Purpose Deletes a specified range of lines in a file.

Syntax [<line>][,<line>]D[Retrn]

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

Example 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 text
27:*in your file.
```

To Delete Multiple Lines To delete lines 5 through 24, type:

5,24D[Retrn]

The result is:

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

To Delete a Single Line

To delete line 6, type:

6D[Retrn]

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.

To Delete a Range of Lines

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

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

To delete a range of lines beginning with the current line 3, type:

,6D[Retrn]

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.

Purpose Edits line of text.

Syntax [<line>][Retrn]

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

If no line number is typed (that is, if only [Retrn] is pressed), the line after the current line (marked with an asterisk (*)) is edited. If no changes to the current line are needed and the cursor is at the beginning or end of the line, press [Retrn] to accept the line as is.

★ ★ CAUTION ★ ★

If [Retrn] is pressed while the cursor is in the middle of the line, the remainder of the line is deleted.

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

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

To edit line 4, type:

4[Retrn]

Example (cont.)

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

```
4:*four.
4:*
```

Now, type:

[Ins]number[space]	4: number
[F3][Retrn]	4: number four.
	5:*

Purpose Ends the editing session.

Syntax E[Retrn]

Comments This command saves the edited file on the drive with the filename and extension you specified in the EDLIN command, renames the original input file with an extension of .BAK, and then exits EDLIN. If the file was created during the editing session, no .BAK file is created.

The E command takes no options. Therefore, you cannot tell EDLIN on which drive to save the file. The drive you want to save the file on must be selected when the editing session is started. If the drive is not selected when EDLIN is started, the file will be saved on the default drive. It will still be possible to COPY the file to a different drive using the DOS COPY command.

You must be sure that the drive contains enough free space for the entire file. If the drive does not contain enough free space, the write will be canceled and the edited file lost, although part of the file might be written out to the drive.

Example Type:

E[Retrn]

After execution of the E command, the DOS default drive prompt (for example, C>) is displayed.

Purpose Inserts text immediately before the specified <line>.

Syntax [<line>]I[Retrn]

Comments If you are creating a new file, the I command must be given before text can be typed (inserted). Text begins with line number 1. Successive line numbers appear automatically each time [Retrn] is pressed.

EDLIN remains in insert mode until [Ctrl]C is typed, or until [Ctrl]Z is typed at any point after column 1. (Pressing [Ctrl]Z in column 1 ends the file.) When the insert is completed and insert mode has been exited, the line immediately following the inserted lines becomes the current line. All line numbers following the inserted section are incremented by the number of lines inserted.

If <line> is not specified, the default will be the current line number and the lines will be inserted immediately before the current line. If <line> is any number larger than the last line number, or if a pound sign (#) is specified as <line>, the inserted lines will be appended to the end of the file. In this case, the last line inserted will become the current line.

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

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

Examples (cont.)

To insert text before a specific line, type <**line**>**I[Retrn]** as in this example:

7I[Retrn]

The result is:

7:*

Now, type the new text for line 7:

7:***and renumber lines[Retrn]**

End Insert

Then to end the insertion, press **[Ctrl]Z** on the next line:

8:***[Ctrl]Z[Retrn]**

Now type **L[Retrn]** 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.

Insert Before Current Line

To insert lines immediately before the current line, type:

I[Retrn]

The result is:

8:*

Now, insert the following text and terminate with a [Ctrl]Z on the next line:

8:***so they are consecutive[Retrn]**
9:***[Ctrl]Z[Retrn]**

Insert Before Current Line (cont.)

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

L[Retrn]

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 append new lines to the end of the file, type:

10I[Retrn]

This produces the following:

10:*

Now, type the following new lines:

10:***The insert command can place new lines[Retrn]**
11:***in the file; there's no problem[Retrn]**
12:***because the line numbers are dynamic[Retrn]**;
13:***they'll go all the way to 65533.[Retrn]**

To Append New Lines (cont.)

End the insertion by pressing [Ctrl]Z on line 14. The new lines will appear at the end of all previous lines in the file. Now type the List command:

L[Retrn]

The result is:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: and renumber lines
 8: so they are consecutive
 9: in your file.
10: The insert command can place new lines
11: in the file; there's no problem
12: because the line numbers are dynamic;
13: they'll go all the way to 65533.
```

Purpose Lists a range of lines, including the two lines specified.

Syntax [<line>][,<line>]L[Retrn]

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[Retrn]

the display will start 11 lines before the current line and end with the specified <line>. The beginning comma is required to indicate the omitted first option.

Note: If the specified <line> is more than 11 lines before the current line, the display will be the same as if you omitted both options.

If you omit the second option, as in

<line>L[Retrn]

23 lines will be displayed, starting with the specified <line>.

If you omit both parameters, as in

L[Retrn]

23 lines will be displayed—the 11 lines before the current line, the current line, and the 11 lines after the current line. If there are less than 11 lines before the current line, more than 11 lines after the current line will be displayed to make a total of 23 lines.

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

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

To list a range of lines without reference to the current line, type <line>,<line>L and press [Retrn]:

2,5L[Retrn]

The result is:

```
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
```

To list a range of lines beginning with the current line, type ,<line> L and press [Retrn]:

,26L[Retrn]

The result is:

```
15:*The current line contains an asterisk.
  .
  .
  .
26: to edit text
```

Examples (cont.)

To list a range of 23 lines centered around the current line, type only L and press [Retrn]:

L[Retrn]

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.
```

Purpose Moves a range of text to the line specified.

Syntax [<line>],[<line<], <line>M[Retrn]

Comments Use the Move command to move a block of text (from the first <line> to the second <line>) to another location in the file. The lines are renumbered according to the direction of the move. For example,

,+25,100M[Retrn]

moves the text from the current line plus 25 lines to line 100. If the line numbers overlap, EDLIN will display an "Entry error" message.

To move lines 20-30 to line 100, type:

20,30,100M[Retrn]

EDLIN COMMANDS
(P)age Command

Purpose Pages through file 23 lines at a time.

Syntax [<line>][,<line>]P[Retrn]

Comments If the first <line> is omitted, that number will default to the current line plus one. If the second <line> is omitted, 23 lines will be listed. The new current line becomes the last line displayed and is marked with an asterisk.

Purpose Quits the editing session, does not save any editing changes, and exits to DOS.

Syntax Q[Retrn]

Comments EDLIN prompts you to make sure you don't want to save the changes.

Type **Y** if you want to quit the editing session. No editing changes are saved and no .BAK file is created. Refer to the End command in this chapter for information about the .BAK file.

Type **N** or any other character except Y if you want to continue the editing session.

Note: When started, EDLIN erases any previous copy of the file with an extension of .BAK to make room to save the new copy. If you reply Y to the Abort edit (Y/N)? message, your previous backup copy will no longer exist.

Example

```
Q
Abort edit (Y/N)?Y
C>
```

Purpose Replaces all occurrences of a string of text in the specified range with a different string of text or blanks.

Syntax [<line>][,<line>][?]R[<string1>][[Ctrl]Z<string2>][Retrn]

Comments As each occurrence of <string1> is found, it is replaced by <string2>. Each line in which a replacement occurs will be displayed. If a line contains two or more replacements of <string1> with <string2>, then the line will be displayed once for each occurrence. When all occurrences of <string1> in the specified range are replaced by <string2>, the R command terminates and the asterisk prompt reappears.

If a second string is to be given as a replacement, then <string1> must be separated from <string2> with a [Ctrl]Z <String2> must also be ended with a [Ctrl]Z [Retrn] combination or with a simple [Retrn].

Omitting Strings If <string1> is omitted, then Replace will take the old <string1> as its value. If there is no old <string1>, i e., this is the first replace done, then the replacement process will be terminated immediately. If <string2> is omitted, then <string1> may be ended with a [Retrn].

If the first <line> is omitted in the range argument (as in ,<line>) then the first <line> will default to the line after the current line If the second <line> is omitted it will default to #. Therefore, this is the same as <line>,#. Remember that # indicates the line after the last line of the file

Omitting Strings (cont.)

If <string1> is ended with a [Ctrl]Z and there is no <string2>, <string2> will be taken as an empty string and will become the new replace string. For example,

R<string1>[Ctrl]Z[Retrn]

will delete occurrences of <string1>, but

R<string1>[Retrn]

will replace <string1> with the old <string2> and

R[Retrn]

will recall the last values entered as <string1> and <string2>. Note that "old" here refers to a previous string specified either in a Search or a Replace command.

? Option

If the question mark (?) option is given, the Replace command will stop at each line with a string that matches <string1>, display the line with <string2> in place, and then display the prompt "O.K.?." If you press Y or the [Retrn] key, then <string2> will replace <string1>, and the next occurrence of <string1> will be found. Again, the O.K.? prompt will be displayed. This process will continue until the end of the range or until the end of the file. After the last occurrence of <string1> is found, EDLIN displays the asterisk prompt.

If you press any key besides Y or [Retrn] after the O.K.? prompt, the <string1> will be left as it was in the line, and Replace will go to the next occurrence of <string1>. If <string1> occurs more than once in a line, each occurrence of <string1> will be replaced individually, and the O.K.? prompt will be displayed after each replacement. In this way, only the desired <string1> will be replaced, and you can prevent unwanted substitutions.

Examples Assume that the following file exists and is ready for editing:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: in your file.
 8: The insert command can place new lines
 9: in the file; there's no problem
10: because the line numbers are dynamic;
11: they'll go all the way to 65533.
```

To replace all occurrences of <string1> with <string2> in a specified range, type:

2,12Rand[Ctrl]Z or[Retrn]

The result is:

```
4: Delete or Insert
5: (the D or I commors)
8: The insert commor can place new lines
```

Note: In the above replacement, some unwanted substitutions have occurred. To avoid these and to confirm each replacement, the same original file can be used with a slightly different command.

In the next example, to replace only certain occurrences of <string1> with <string2>, type:

2?Rand[Ctrl]Z or[Retrn]

Examples (cont.)

The result is:

```
4: Delete or Insert
O.K.? Y
5: (The D or I commands)
O.K.? Y
5: (The D or I commors)
O.K.? N
8: The insert commor can place new lines
O.K.? N
*
```

Now, type the List command

L[Retrn]

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
.
.
.
```

Purpose

Searches the specified range of lines for a specified string of text.

Syntax

[<line>][,<line>][?]S[<string>][Retrn]

Comments

The <string> must be ended with a [Retrn]. The first line that matches <string> is displayed and becomes the current line. If the question mark option is not specified, the Search command will terminate when a match is found. If no line contains a match for <string>, the message "Not found" will be displayed.

? Option

If the question mark option (?) is included in the command, EDLIN will display the first line with a matching string; it will then prompt you with the message "O.K.?." If you press either the Y or [Retrn] key, the line will become the current line and the search will terminate. If you press any other key, the search will continue until another match is found, or until all lines have been searched (and the "Not found" message is displayed).

Omitting Strings

If the first <line> is omitted (as in ,<line>S<string>), the first <line> will default to the line after the current line. If the second <line> is omitted (as in <line>S<string> or <line>,S<string>), the second <line> will default to # (line after last line of file), which is the same as <line>,# S<string>. If <string> is omitted, Search will take the old string if there is one. (Note that "old" here refers to a string specified in a previous Search or Replace command.) If there is not an old string (i.e., no previous search or replace has been done), the command will terminate immediately.

Examples Assume that the following file exists and is ready for editing:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: in your file.
 8: The insert command can place new lines
 9: in the file; there's no problem
10: because the line numbers are dynamic;
11:*they'll go all the way to 65533.
```

To search for the first occurrence of the string "and", type:

2,12Sand[Retrn]

The following line is displayed:

4: Delete and Insert

To get the "and" in line 5, modify the search command by typing:

[Del][F3],12Sand[Retrn]

The search then continues from the line after the current line (line 4), since no first line was given. The result is:

5: (the D and I commands)

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

1,?Sand[Retrn]

The result is:

```
4: Delete and Insert
O.K.?
```

Examples (cont.)

If you press any key (except Y or [Retrn]), the search continues, so type N here:

O.K.? **N**

Continue:

5: (the D and I commands)
O.K.?

Now press Y to terminate the search:

O.K.? **Y**
*

To search for string XYZ without the verification (O.K.?), type:

SXYZ[Retrn]

EDLIN will report a match and will continue to search for the same string when you issue the S command:

S[Retrn]

EDLIN reports another match.

S[Retrn]

EDLIN reports the string is not found.

Note: <string> defaults to any string specified by a previous Replace or Search command.

Purpose Inserts (merges) the contents of <filename> into the file currently being edited at <line>. If <line> is omitted, then the current line will be used.

Syntax [<line>]T[d:]<filename>[Retrn]

Comments This command is useful if you want to put the contents of a file into another file or into the text you are typing. The transferred text is inserted at the line number specified by <line> and the lines are renumbered.

EDLIN COMMANDS
(W)rite 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.

Syntax [<n>]W[Retrn]

Comments This command is meaningful only if the file you are editing is too large to fit into memory. When you start EDLIN, EDLIN reads lines into memory until memory is 3/4 full.

To edit the remainder of your file, you must write edited lines in memory to disk. Then you can load additional unedited lines from disk into memory by using the Append command.

Note: If you do not specify the number of lines, lines will be written until memory is 3/4 full. No action will be taken if available memory is already more than 3/4 full. All lines are renumbered, so that the first remaining line becomes line number 1.

Error Message Display

When EDLIN finds an error, one of the following error messages is displayed:

Error Message	Cause	Cure
Cannot Edit .BAK file--Rename File	You attempted to edit a file with a filename extension of .BAK. The .BAK files cannot be edited because this extension is reserved for backup copies.	If you need the .BAK file for editing purposes, you must either RENAME the file with a different extension or COPY the .BAK file and give it a different filename extension.
No Room In Directory For File	When you attempted to create a new file, either the file directory was full or you specified an illegal drive or an illegal filename.	Check the command line that started EDLIN for illegal filename and illegal drive entries. If the command is no longer on the display and if you have not yet typed a new command, the EDLIN start command can be recovered by pressing [F3] If this command line contains no illegal entries, run the CHKDSK program for the specified drive. If the status report shows that the directory is full, replace the diskette or use another drive.

EDLIN ERROR MESSAGES
(cont.)

Error Message	Cause	Cure
Line Too Long	During a Replace command, the string given as the replacement caused the line to expand beyond the limit of 253 characters. EDLIN canceled the Replace command.	Divide the long line into two lines, then try the Replace command twice.
Disk Full--File Write Not Completed	You gave the End command, but the drive did not contain enough free space for the whole file. EDLIN canceled the E command and returned you to the operating system. Some of the file may have been written to the drive.	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 drive has sufficient free space for the file to be written to the drive before you begin your editing session.
Invalid Drive Name Or File	You have not specified a valid drive or filename when starting EDLIN.	Specify the correct drive or filename.
Filename Must Be Specified	You did not specify a filename when you started EDLIN.	Specify a filename.
Invalid Parameter	You specified a switch other than /B when starting EDLIN.	Specify the /B switch when you start EDLIN.

Error Message	Cause	Cure
Insufficient Memory	There is not enough memory to run EDLIN.	You must free some memory by writing files to another drive or by deleting files before restarting EDLIN.
File Not Found	The filename specified during a Transfer command was not found.	Specify a valid filename when issuing a Transfer command.
Must Specify Destination Number	A destination line number was not specified for a Copy or Move command.	Reissue the command with a destination line number.
Not Enough Room To Merge The Entire File	There was not enough room in memory to hold the file during a Transfer command.	You must free some memory by writing some files to the drive or by deleting some files before you can transfer this file.
File Creating Error	The EDLIN temporary file cannot be created.	Check to make sure that the directory has enough space to create the temporary file. Also, make sure that the file does not have the same name as a subdirectory in the directory where the file to be edited is located.

CHAPTER 7

FILE COMPARISON UTILITY

Overview

It is sometimes useful to compare files on your drives. If you have copied a file and later want to compare copies to see which one is current, you can use the File Comparison Utility (FC).

Note: The commands for the File Comparison Utility are optional and they may not have been installed on your system. Check with your system manager.

The File Comparison Utility compares the contents of two files. The differences between the two files can be output to the display 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 (files output by the MACRO-86 assembler, the MS-LINK Linker utility, or by a Microsoft high-level language compiler).

The comparisons are made in one of two ways: on a line-by-line or a byte-by-byte basis. The line-by-line comparison isolates blocks of lines that are different between the two files and prints those blocks of lines. The byte-by-byte comparison displays the bytes that are different between the two files.

Limitations on Source Comparison

FC uses a large amount of memory as buffer (storage) space to hold the source files. If the source files are larger than available memory, FC will compare what can be loaded into the buffer space. If no lines match in the portions of the files in the buffer space, FC will display only the message:

*** Files are different ***

For binary files larger than available memory, FC compares both files completely, overlaying the portion in memory with the next portion from disk. All differences are output in the same manner as those files that fit completely in memory.

FC Syntax

The syntax of FC is as follows:

FC [/#] [/B] [/W] [/C] <filespec> <filespec>[Retrn]

FC matches the first file against the second and reports any differences between them. For example:

FC D:\FOO\BAR\FILE1.TXT \BAR\FILE2.TXT

FC takes FILE1.TXT in the \FOO\BAR directory of drive D and compares it with FILE2.TXT in the \BAR directory. Since no drive is specified for filename2, FC assumes that the \BAR directory is on the default drive.

FC SWITCHES

Introduction There are four switches that you can use with the File Comparison Utility:

/B Forces a binary comparison of both files. The two files are compared byte-to-byte, with no attempt to re-synchronize after a mismatch. The mismatches are printed as follows:

```
--ADDRS----F1----F2-
xxxxxxxx    yy   zz
```

(where xxxxxxxx is the relative address of the pair of bytes from the beginning of the file). Addresses start at 00000000; yy and zz are the mismatched bytes from file1 and file2, respectively. If one of the files contains less data than the other, then a message is printed out. For example, if file1 ends before file2, then FC displays:

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

/# # stands for a number from 1 to 9. This switch specifies the number of lines required to match for the files to be considered as matching again after a difference has been found. If this switch is not specified, it defaults to 3. This switch is used only in source comparisons.

/W Causes FC to compress whites (tabs and spaces) during the comparison. Thus, multiple contiguous whites in any line will be considered as a single white space. Note that although FC compresses whites, it does not ignore them. The two exceptions are beginning and ending whites in a line, which are ignored. For example, (an underscore represents a white)

_____More____data__to__be__found_____

will match with

More__data__to__be__found

and with

_____More_____data__to__be_____found_____

but will not match with

_____Moredata__to__be__found

This switch is used only in source comparisons.

/C Causes the matching process to ignore the case of letters. All letters in the files are considered uppercase letters. For example,

Much__MORE__data__IS__NOT__FOUND

will match

much__more__data__is__not__found

If both the /W and /C options are specified, then FC will compress whites and ignore case. For example,

_____DATA__was__found_____

will match:

data__was__found

This switch is used only in source comparisons.

Description The File Comparison Utility reports the differences between the two files you specify by displaying the first filename, followed by the lines that differ between the files, followed by the first line to match in both files. FC then displays the name of the second, file followed by the lines that are different, followed by the first line that matches. The default for the number of lines to match between the files is 3. (If you want to change this default, specify the number of lines with the /# switch.) For example:

```
...
...
----------<filename1>
<difference>
<1st line to match file2 in file1>
----------<filename2>
<difference>
<1st line to match file1 in file2>
```

FC will continue to list each difference.

If there are too many differences (involving too many lines), the program will simply report that the files are different and stop.

If no matches are found after the first difference is found, FC will display:

*** Files are different ***

and will return to the DOS default drive prompt (for example, C>).

Description The differences and matches between the two files you specify will be shown on your display unless you redirect the output to a file. This is accomplished in the same way as DOS command redirection (refer to Chapter 3, "Learning About Commands").

To compare File1 and File2 and then send the FC output to DIFFER.TXT, type:

FC File1 File2 >DIFFER.TXT[Retrn]

The differences and matches between File1 and File2 will be put into DIFFER.TXT on the default drive.

EXAMPLES OF FC OUTPUT

Example 1 Assume these two ASCII files are on a drive:

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

To compare the two files and show the differences on the display, type:

FC ALPHA.ASM BETA.ASM[Retrn]

Example 1 (cont.)

FC compares ALPHA.ASM with BETA.ASM and shows the differences on the display. (The defaults are: do not use tabs, spaces, or comments for matches, and do a source comparison on the two files.)

Output

The output will appear as follows on the display (the Notes do not appear):

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

----------ALPHA.ASM
M                          NOTE: ALPHA file contains
N                          MNO where BETA contains
O                          J12.
P

----------BETA.ASM
J
1
2
P
--------------------

----------ALPHA.ASM
W                          NOTE: ALPHA file contains
                           W where BETA contains 45W.
----------BETA.ASM
4
5
W
```

Example 2 You can print the differences on the line printer using the same two source files. In this example, four successive lines must be the same to constitute a match.

Type:

FC /4 ALPHA.ASM BETA.ASM >**PRN[Retrn]**

Output The following output will be sent to your default printer:

```
----------ALPHA.ASM
D
E
F
G
H
I
M
N                          NOTE: P is the first of a
P                          string of four matches.

----------BETA.ASM
G
H
I
J
1
2
P

------------------

----------ALPHA.ASM
W                          NOTE: W is the first of a
----------BETA.ASM         string of four matches.
4
5
W
```

Example 3 This example forces a binary comparison and then shows the differences on the display using the same two source files as were used in the previous examples.

Type:

FC /B ALPHA.ASM BETA.ASM[Retrn]

Display The /B switch in this example forces binary comparison. This switch and any others must be typed before the filenames in the FC command line. The following display should appear:

```
--ADDRS----F1---F2--
 00000009  44  47
 0000000C  45  48
 0000000F  46  49
 00000012  47  4A
 00000015  48  31
 00000018  49  32
 0000001B  4D  50
 0000001E  4E  51
 00000021  4F  52
 00000024  50  53
 00000027  51  54
 0000002A  52  55
 0000002D  53  56
 00000030  54  34
 00000033  55  35
 00000036  56  57
 00000039  57  58
 0000003C  58  59
 0000003F  59  5A
 00000042  5A  1A
```

Error Message Display

When the File Comparison Utility detects an error, one or more of the following error messages will be displayed:

Error Message	Cause
Invalid parameter: <option>	One of the switches that you have specified is invalid.
File not found: <filename>	FC could not find the filename you specified.
Read error in: <filename>	FC could not read the entire file.
Invalid number of parameters	You have not specified two files to be compared.

CHAPTER 8

THE LINKER PROGRAM (MS-LINK)

Overview

If you want to compile and link programs, you can use the linker program, called MS-LINK. This chapter explains how to use it.

The DOS linker (called MS-LINK) 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.

Note: MS-LINK commands are optional and they may not have been installed on your system. Check with your system manager.

Description

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.

Description (cont.)

MS-LINK combines several object modules into one relocatable load module, or Run file (also called a .EXE or Executable file). As it combines modules, MS-LINK makes sure that all external references between object modules are defined. MS-LINK can search several library files for definitions of any external references that are not defined in the object modules.

MS-LINK produces a List file that shows external references resolved, and it also displays any error messages.

MS-LINK uses available memory as much as possible. When available memory is exhausted, MS-LINK creates a temporary file named VM.TMP.

DOS Memory Divisions

Some of the terms used in this chapter are explained below to help you understand how MS-LINK works. Generally, if you are linking object modules compiled from BASIC, Pascal, or a high-level language, you will not need to know these terms. If you are writing and compiling programs in assembly language, however, you will need to understand MS-LINK and the definitions described below. The section MS-LINK in the Macro Assembler Manual also contains useful information on how MS-LINK works.

In DOS, memory can be divided into segments, classes, and groups.

Example

Segment Number	Segment Name	Segment Class Name
Segment 1	PROG.1	CODE
Segment 2	PROG.2	CODE
Segment 12	PROG.3	DATA

Note: 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 (i.e , the lowest address in memory).

Each segment has a segment name and a class name. MS-LINK 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, MS-LINK references segments by their addresses in memory (where they are located). MS-LINK does this by finding groups of segments.

DOS Groups

A group is a collection of segments that fit within a 64K byte area of memory. The segments do not need to be contiguous to form a group. The address of any group is the lowest address of the segments in that group. At link time, MS-LINK 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.

MS-LINK

- ☆ 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

Input File Extensions

If no filename extensions are given in the input (object) file specifications, MS-LINK will recognize the following extensions by default:

.OBJ	Object
.LIB	Library

Output File Extensions

MS-LINK appends the following default extensions to the output (Run and List) files:

.EXE	Run (may not be overridden)
.MAP	List (may be overridden)

VM.TMP (Temporary) File

MS-LINK uses available memory for the link session. If the files to be linked create an output file that exceeds available memory, MS-LINK will create a temporary file, name it VM.TMP, and put it on the default drive. If MS-LINK creates VM.TMP, it will display the message:

```
VM.TMP has been created.
Do not change diskette in drive, <d:>
```

If you are using a diskette, 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 MS-LINK will be unpredictable, and MS-LINK might display the error message:

```
Unexpected end of file on VM.TMP
```

VM.TMP (Temporary) File (cont.)

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.

★ ★ CAUTION ★ ★

Do not use VM.TMP as a filename for any file. If you have a file named VM.TMP on the default drive and MS-LINK requires the VM.TMP file, MS-LINK will delete the VM.TMP already on disk and create a new VM.TMP. Thus, the contents of the previous VM.TMP file will be lost.

Types of Input

MS-LINK requires two types of input: a command to start MS-LINK and responses to command prompts. In addition, seven switches control MS-LINK features. Usually, you will type all the commands to MS-LINK on the keyboard. As an option, answers to the command prompts and any switches may be contained in a response file. Command characters can be used to assist you while giving commands to MS-LINK.

Methods of Starting

MS-LINK may be started in any of three ways. The first method is to type the commands in response to individual prompts. In the second method, you type all commands on the line used to start MS-LINK. To start MS-LINK by the third method, you must create a response file that contains all the necessary commands and tell MS-LINK where that file is when you start MS-LINK.

Summary of Methods to Start MS-LINK:

Method 1	LINK
Method 2	LINK <filenames>[/switches]
Method 3	LINK @<filespec>

HOW TO START MS-LINK
Prompts

Method 1: Prompts

To start MS-LINK with Method 1, type:

LINK[Retrn]

MS-LINK will be loaded into memory MS-LINK will then display four text prompts that appear one at a time. You answer the prompts to command MS-LINK to perform specific tasks

At the end of each line, you may type one or more switches, preceded by the switch character, a slash.

The command prompts are summarized below and described in more detail in the ''Command Prompts'' section

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 typed, the prompt will reappear. There is no default; a response is required.
Run File [Object-file.EXE]:	Give filename for executable object code The default is the first object-filename.EXE. (You cannot change the output extension.)
List File [Run-file.MAP]:	Give filename for listing. The default is NUL.MAP. If you want to create a list file, you must specify a filename
Libraries [ ]	List filenames to be searched, separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear. The default will search for default libraries in the object modules. (Extensions will be changed to LIB.)

Method 2: Command Line

To start MS-LINK using Method 2, type all commands on one line. The entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas. Use the following syntax.

LINK <object-list>,<runfile>,<listfile>,<lib-list>[/switch][Retrn]

where: *object-list* is a list of object module files, separated by plus signs

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

/switch refers to optional switches, which may be placed following any of the response entries (just before any of the commas or after the <lib-list>, as shown)

The entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas.

To select the default for a field, simply type a second comma with no spaces between the two commas.

Example

LINK FUN+TEXT+TABLE+CARE/P/M,,FUNLIST,COBLIB.LIB[Retrn]

This command causes MS-LINK to be loaded, then the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ are loaded. MS-LINK then pauses (as a result of using the /P switch). MS-LINK links the object modules when you press any key, and produces a global symbol map (the /M switch); 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 MS-LINK with Method 3, use the following syntax:

LINK @<filespec>[Retrn]

where: *filespec* is the name of a response file. A response file contains answers to the MS-LINK prompts (shown in Method 1) and may also contain any of the switches. When naming a response file, the use of filename extensions is optional. Method 3 permits the command that starts MS-LINK 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 an MS-LINK prompt. The responses must be in the same order as the MS-LINK prompts discussed in Method 1. If desired, a long response to the Object Modules: or Libraries: prompt may be typed on several lines by using a plus sign (+) to continue the same response onto the next line.

Use switches and command characters in the response file the same way as they are used for responses typed on the keyboard.

When the MS-LINK session begins, each prompt will be displayed in order with the responses from the response file. If the response file does not contain answers for all the prompts, (in the form of filenames, the semicolon command character or [Retrn]), MS-LINK will display the prompt which does not have a response, then wait for you to type a legal response. When a legal response has been typed, MS-LINK continues the link session.

Method 3: Response File (cont.)

```
FUN TEXT TABLE CARE
/PAUSE/MAP
FUNLIST
COBLIB.LIB
```

This response file tells MS-LINK to load the four object modules named FUN, TEXT, TABLE, and CARE. MS-LINK pauses before producing a public symbol map to permit you to swap disks (see discussion under /PAUSE in the "Switches" section before using this feature). When you press any key, the output files will be named FUN.EXE and FUNLIST.MAP. MS-LINK will search the library file COBLIB.LIB, and will use the default settings for the switches.

Plus Sign

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 type a large number of responses (each may be very long), type a plus sign and a [Retrn] at the end of the line to extend it. If the plus sign and a [Retrn] is the last entry following these two prompts, MS-LINK will prompt you for more module names. When the Object Modules: or Libraries: prompt appears again, continue to type 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 [Retrn] and not a plus sign and a [Retrn]. For example:

```
Object Modules [.OBJ]: FUN TEXT TABLE CARE+[Retrn]
Object Modules [.OBJ]:FOO+FLIPFLOP+JUNQUE+[Retrn]
Object Modules [.OBJ]:CORSAIR[Retrn]
```

Semicolon

To select default responses to the remaining prompts, use a single semicolon (;) followed immediately by [Retrn] at any time after the first prompt (Run File:). This feature saves time and overrides the need to press [Retrn] for each prompt.

Note: Once the semicolon has been typed and entered (by pressing [Retrn]), you can no longer respond to any of the prompts for that link session. Therefore, do not use the semicolon to skip prompts. To skip prompts, use [Retrn]. For example, if you use

```
Object Modules [.OBJ]: FUN TEXT TABLE CARE[Retrn]
Run Module [FUN.EXE]: ;[Retrn]
```

no other prompts will appear, and MS-LINK will use the default values.

[Ctrl]C Use [Ctrl]C to cancel the link session at any time. If you type an erroneous response, such as the wrong filename or an incorrectly spelled filename, you must press [Ctrl]C to exit MS-LINK, and then restart MS-LINK. If the error has been typed but you have not pressed [Retrn], you may delete the erroneous characters with [Del], but for that line only.

COMMAND PROMPTS

Introduction

MS-LINK asks you for responses to four text prompts. When you have typed a response to a prompt and pressed [Retrn], the next prompt appears. When the last prompt has been answered, MS-LINK begins linking automatically without further command. When the link session is finished, MS-LINK exits to the operating system. When the operating system prompt appears, MS-LINK has finished successfully. If the link session is unsuccessful, MS-LINK will display the appropriate error message.

MS-LINK 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, for prompts which can default to preset responses.

Object Modules Prompt

The Object Modules: prompt has no preset filename response and requires you to type a filename.

Type a list of the object modules to be linked. MS-LINK assumes by default that the filename extension is .OBJ. If an object module has any other filename extension, the extension must be given. Otherwise, the extension may be omitted.

Modules must be separated by plus signs (+).

Remember that MS-LINK loads segments into classes in the order encountered. You can use this information to set the order in which the object modules will be read by MS-LINK. Refer to the Macro Assembler Manual for more information on this process.

Run File Prompt

Typing a filename will create a file for storing 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 typed to the Run File: prompt, MS-LINK uses the first filename typed in response to the Object Modules: prompt as the RUN filename. For example:

Run File [FUN.EXE]: A:PAYROLL

This response directs MS-LINK to create the Run file PAYROLL.EXE on drive A.

List File Prompt

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 NULL.MAP

Libraries Prompt

The valid responses are up to eight library filenames or simply a [Retrn]. (A [Retrn] means default library search.) Library files must have been created by a library utility. (Consult the MS-LIB section of the Macro Assembler Manual for information on library files.) MS-LINK assumes by default that the filename extension is .LIB for library files.

Library filenames must be separated by blank spaces or plus signs (+).

MS-LINK searches library files in the order listed to resolve external references. When it finds the module that defines the external symbol, MS-LINK processes that module as another object module.

If MS-LINK cannot find a library file on the drives, it will display the message:

Cannot find library <library-name> Type new drive letter:

Press the letter for the drive (for example, D).

Introduction The seven MS-LINK switches control various MS-LINK functions. Switches must be typed at the end of a prompt response, regardless of which method is used to start MS-LINK. Switches may be grouped at the end of any response, or may be scattered at the end of several. If more than one switch is typed at the end of one response, each switch must be preceded by a slash (/).

All switches may be abbreviated. The only restriction is that an abbreviation must be sequential from the first letter through the last typed; no gaps or transpositions are allowed. For example, in the /DSALLOCATE switch:

Legal	Illegal
/D	/DSL
/DS	/DAL
/DSA	/DLC
/ DSALLOCA	/DSALLOCT

/DSALLOCATE Using the /DSALLOCATE switch tells MS-LINK to load all data at the high end of the data segment (DS). Otherwise, MS-LINK loads all data at the low end of the data segment. At runtime, the DS pointer is set to the lowest possible address to allow the entire DS segment to be used. Use of the /DSALLOCATE switch in combination with the default load low (that is, the /HIGH switch is not used) permits the user application to dynamically allocate any available memory below the area specifically allocated within the group of data segments, yet to remain addressable by the same DS pointer. This dynamic allocation is needed for Pascal and FORTRAN programs.

Note: Your application program may dynamically allocate up to 64K bytes (or the actual amount of memory available) less the amount allocated within the group of data segments.

/HIGH

Use of the /HIGH switch causes MS-LINK to place the Run file as high as possible in memory. Otherwise, MS-LINK places the Run file as low as possible.

★ ★ CAUTION ★ ★

Do not use the /HIGH switch with Pascal or FORTRAN programs.

/LINE-NUMBERS

The /LINENUMBERS switch tells MS-LINK to include in the List file the line numbers and addresses of the source statements in the input modules. Otherwise, line numbers are not included in the List file.

Note: Not all compilers produce object modules that contain line number information. In these cases, of course, MS-LINK cannot include line numbers.

/MAP

/MAP directs MS-LINK to list all public (global) symbols defined in the input modules. If /MAP is not given, MS-LINK will list only errors (including undefined globals).

The symbols are listed alphabetically. For each symbol, MS-LINK lists its value and its segment:offset location in the Run file. The symbols are listed at the end of the List file.

/PAUSE

The /PAUSE switch causes MS-LINK to pause in the link session when the switch is encountered. Normally, MS-LINK performs the linking session from beginning to end without stopping. This switch allows you to swap diskettes (if you're using diskettes) before MS-LINK outputs the Run (.EXE) file.

When MS-LINK enounters the /PAUSE switch, it displays the message:

About to generate .EXE file
Change diskettes <hit any key>

MS-LINK resumes processing when you press any key.

★ ★ CAUTION ★ ★

Do not remove the diskette which will receive the List file, or the diskette used for the VM.TMP file, if one has been created.

/STACK: <number>

Number represents any positive numeric value (in hexadecimal radix) up to 65536 bytes. If a value from 1 to 511 is typed, MS-LINK will use 512. If the /STACK switch is not used for a link session, MS-LINK will calculate 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, MS-LINK will display the following error message:

Warning: no stack statement

/NO /NO is short for NODEFAULTLIBRARYSEARCH. This switch tells MS-LINK to not search the default (product) libraries in the object modules. For example, if you are linking object modules in Pascal, specifying the /NO switch tells MS-LINK to not automatically search the library named PASCAL.LIB to resolve external references.

Introduction This sample shows you the type of information that is displayed during an MS-LINK session.

In response to the prompt, type:

LINK[Retrn]

This is an example of the messages and prompts the system might display (user responses are printed in boldface):

Microsoft Object Linker V.2.xx(C) Copyright 198x by Microsoft Inc.

Object Modules [.OBJ]: **IO PROG[Retrn]**
Run File [IO.EXE]:**[Retrn]**
List File [NUL.MAP]: **IO /MAP[Retrn]**
Libraries [.LIB]: **;[Retrn]**

Note: By specifying /MAP, you get both an alphabetic listing and a chronological listing of public symbols.

By responding PRN to the List File: prompt, you can redirect your output to the printer.

By specifying the /LINE switch, MS-LINK gives you a listing of all line numbers for all modules. (Note that the /LINE switch can generate a large volume of output.)

By pressing [Retrn] in response to the Libraries: prompt, an automatic library search is performed.

Linker Map

Once MS-LINK locates all libraries, the linker map can display a list of segments in the order of their appearance within the load module. Type:

IO.MAP[Retrn]

The result is:

Start	Stop	Length	Name	Class
00000H	00001H	0002H	BASE	PROG
00002H	04495H	4494H	PROG	PROG
044A0H	06C6BH	27CCH	DATA	DATA
06C70H	06D6FH	0100H	STACK	DATA

Origin	Group
044A:0	DGROUP
0000:0	PGROUP

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. Consult the Macro Assembler and Utilities Manual for information on how to determine where relative zero is actually located, and also on how to determine the absolute address of a segment.

Public Symbols

Because the /MAP switch was used, MS-LINK displays the public symbols by name and value. For example:

Address	Publics By Name
044A:254C	ARGC
0000:0002	C
0000:0617	GET_UIT
0000:04EA	HEAD_QUE
044A:046E	LOCAL_QU
0000:1F29	MOVMEN
0000:1CE6	READ_UIT
0000:0136	XCOVF

Address	Publics By Value
0000:0002	C
0000:0136	XCOVF
0000:04EA	HEAD_QUE
0000:0617	GET_UIT
0000:1CE6	READ_UIT
0000:1F29	MOVMEN
044A:046E	LOCAL_QU
044A:254C	ARGC

MS-LINK ERROR MESSAGES

Error Message Display

All errors cause the link session to be canceled After the cause has been found and corrected, MS-LINK must be re-run. The following error messages are displayed by MS-LINK:

Error Message	Explanation
Attempt to access data outside of segment bounds possibly bad object module	There is probably a bad Object file.
Bad numeric parameter	Numeric value is not in digits.
Cannot open temporary file	MS-LINK is unable to create the file VM.TMP because the disk directory is full. Delete other unneeded files on the diskette or in the drive to make more space.
Error: DUP record too complex	DUP record in assembly language module is too complex. Simplify DUP record in assembly language program.
Error: fixup offset exceeds field width	An assembly language instruction refers to an address with a short instruction instead of a long instruction Edit assembly language source and reassemble.
Input file read error	There is probably a bad Object file.
Invalid object module	An object module(s) is incorrectly formed or incomplete (as when assembly is stopped in the middle)

Error Message	Explanation
Symbol defined more than once	MS-LINK found two or more modules that define a single symbol name.
Program size or number of segments exceeds capacity of linker	The total size may not exceed 384K bytes and the number of segments may not exceed 255.
Requested stack size exceeds 64K	Specify a size greater than or equal to 64K bytes with the /STACK switch.
Segment size exceeds 64K	64K bytes is the addressing system limit.
Symbol table capacity exceeded	Very many and/or very long names were typed, exceeding the limit of approximately 25K bytes.
Too many external symbols in one module	The limit is 256 external symbols per module.
Too many groups	The limit is 10 groups.
Too many libraries specified	The limit is 8 libraries.
Too many public symbols	The limit is 1024 public symbols.
Too many segments or classes	The limit is 256 (segments and classes taken together).
Unresolved externals: <list>	The external symbols listed have no defining module among the modules or library files specified.

Error Message	Explanation
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.
VM read error	This is a disk error; it is not caused by MS-LINK.
Warning: no stack segment	None of the object modules specified contains a statement allocating stack space, but you typed the /STACK switch.
Warning: segment of absolute or unknown type	There is a bad object module or an attempt has been made to link modules that MS-LINK cannot handle (e.g., an absolute object module).
Write error in TMP file	No more space remains to expand the VM.TMP file.
Write error on run file	Usually, there is not enough space for the Run file.

CHAPTER 9

DEBUG PROGRAM

Overview of DEBUG

The DEBUG program (DEBUG) 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 re-execute a program to check on the validity of the changes.

All DEBUG commands may be cancelled at any time by pressing [Ctrl]C. [Ctrl]S suspends the display, so that you can read it before the output scrolls away. Entering any key other than [Ctrl]C or [Ctrl]S restarts the display. All of these commands are consistent with the control character functions available at the DOS command level.

HOW TO START DEBUG

Methods

DEBUG may be started two ways. By the first method, you type all commands in response to the DEBUG prompt (a hyphen) By the second method, you type all commands on the line used to start DEBUG

Method 1: DEBUG

To start DEBUG using method 1, type.

DEBUG[Retrn]

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 files can be worked on by using other commands.

★ ★ CAUTION ★ ★

When DEBUG is started, it sets up a program header at offset 0 in the program work area. On previous versions of DEBUG, you could over-write this header. You can still overwrite the default header if no <filespec> is given to DEBUG. If you are debugging a .COM or .EXE file, however, do not tamper with the program header below address 5CH, or DEBUG will terminate.

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

The syntax for DEBUG using a command line, is·

DEBUG [<filespec>][<arglist>][Retrn]

For example, the following is a typical command to start DEBUG:

DEBUG FILE EXE[Retrn]

Method 2: Command Line (cont.)

DEBUG then loads FILE.EXE into memory starting at 100 hexadecimal in the lowest available segment. The BX:CX registers are loaded with the number of bytes placed into memory.

An <arglist> may be specified if a <filespec> is present. The <arglist> is a list of filename parameters and switches that are to be passed to the program <filespec>. Thus, when <filespec> is loaded into memory, it is loaded as if it had been started with the command:

<filespec> <arglist>

Here, <filespec> is the file to be debugged, and the <arglist> is the rest of the command line that is used when <filespec> is invoked and loaded into memory.

COMMAND INFORMATION

Introduction Each DEBUG command consists of a single letter followed by one or more parameters. Additionally, the control characters and the special editing functions described in earlier chapters of this manual apply inside DEBUG.

If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with a (^) 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.

Table of DEBUG Commands

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

COMMAND INFORMATION
Parameters

Delimiter

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 of Parameters

Parameter	Definition
<drive>	A one-digit hexadecimal value to indicate which drive a file will be loaded from or written to. The valid values are 0-F hex These values designate the drives as follows: 0=A, 1=B, 2=C, 3=D, .0F=P
<byte>	A two-digit hexadecimal value to be placed in or read from an address or register
<record>	A 1- to 3-digit hexadecimal value used to indicate the logical record number on the disk and the number of disk sectors to be written or loaded Logical records correspond to sectors. However, their numbering differs since they represent the entire disk space
<value>	A hexadecimal value up to four digits used to specify a port number or the number of times a command should repeat its functions

Table of Parameters (cont.)

Parameter	Definition
<address>	A two-part designation consisting of either an alphabetic segment register designation or a four-digit segment address plus an offset value. The segment designation or segment address may be omitted, in which case the default segment is used. DS is the default segment for all commands except G, L, T, U, and W, for which the default segment is CS. All numeric values are hexadecimal. For example: CS:0100 04BA:0100 The colon is required between a segment designation (whether numeric or alphabetic) and an offset.

Table of Parameters (cont.)

Parameter	Definition
<range>	<address> <address>; or <address> [L <value>] where <value> is the number of lines the command should operate on, and L80 is assumed. The L and value cannot be omitted if another hex value follows the <range>, since the hex value would be interpreted as the second <address> of the <range>. Examples: CS:100 110 CS:100 L 10 CS:100 The following is illegal: CS:100 CS:110 ^Error The limit for <range> is 10000 hex. To specify a <value> of 10000 hex within four digits, type 0000 (or 0).
<list>	A series of <byte> values or of <string>s. <list> must be the last parameter on the command line. For example: fcs·100 42 45 52 54 41

Table of Parameters (cont.)

Parameter	Definition
<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 string is legal: 'This "string" is okay.' However, this string is illegal: 'This 'string' is not ' Similarly, these strings are legal: "This 'string' is okay." "This z ""'string'"" is okay." However, this string is illegal: "This "string" is not." **Note:** The ASCII values of the characters in the string can be used as a <list> of byte values.

Purpose Assembles 8086/8087/8088 mnemonics directly into memory.

Syntax A[<address>][Retrn]

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 entered as 1-4 characters. Prefix mnemonics must be specified in front of the opcode to which they refer. They may also be entered on a separate line.

The segment override mnemonics are CS:, DS:, ES:, and SS:. The mnemonic for the far return is RETF. String manipulation mnemonics must explicitly state the string size. For example, use MOVSW to move word strings and MOVSB to move byte strings.

The assembler will automatically assemble short, near or far jumps and calls, depending on byte displacement to the destination address. These may be overridden with the NEAR or FAR prefix. For example:

```
0100:0500 JMP 502          ; a 2-byte short jump
0100:0502 JMP NEAR 505     ; a 3-byte near jump
0100:0505 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]
```

Comments (cont.)

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
```

Pseudo-Instructions

Two popular pseudo-instructions are available with Assemble. The DB opcode will assemble byte values directly into memory. The DW opcode will assemble word values directly into memory. For example:

```
DB      1,2,3,4,''THIS IS AN EXAMPLE''
DB      'THIS IS A QUOTE: '' '
DB      ''THIS IS A QUOTE:' ''

DW      1000,2000,3000,''BACH''
```

Assemble supports all forms of register indirect commands. For example:

```
ADD     BX,34[BP+2].[SI-1]
POP     [BP+DI]
PUSH    [SI]
```

All opcode synonyms are also supported. For example:

```
LOOPZ   100
LOOPE   100

JA      200
JNBE    200
```

For 8087 opcodes, the WAIT or FWAIT must be explicitly specified. For example:

```
FWAIT FADD ST,ST(3)  , This line will assemble
                     , an FWAIT prefix
LD TBYTE PTR [BX]    ; This line will not
```

(C)ompare Command

Purpose Compares the portion of memory specified by <range> to a portion of the same size beginning at <address>[Retrn].

Syntax C<range> <address>[Retrn]

Comments If the two areas of memory are identical, there are no display and DEBUG returns with the DOS prompt. If there are differences, they are displayed in this format:

<address1> <byte1> <byte2> <address2>

Example The following commands have the same effect:

C100,1FF 300

or

C100L100 300

Each command compares the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH.

Purpose Displays the contents of the specified region of memory.

Syntax D[<range>]

Comments If a range of addresses is specified, the contents of the range are displayed. If the D command is typed without parameters, 128 bytes are displayed at the first address (DS:100) after the address displayed by the previous Dump command.

The dump is displayed in two portions: a hexadecimal dump (each byte is shown in hexadecimal value) and an ASCII dump (the bytes are shown in ASCII characters). Nonprinting characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows 16 bytes with a hyphen between the eight and and ninth bytes. At times, displays are split in this manual to fit them on the page. Each displayed line begins on a 16-byte boundary.

If you type the command:

DCS:100 110[Retrn]

DEBUG displays the dump in the following format:

04BA:0100 42 45 52 54 41 ... 4E 44 TOM SAWYER

If you type the following command:

D[Retrn]

the display is formatted as described above. Each line of the display begins with an address, incremented by 16 from the address on the previous line. Each subsequent D (typed without parameters) displays the bytes immediately following those last displayed.

Comments If you type the command:

DCS:100 L 20[Retrn]

the display is formatted as described above, but 20H bytes are displayed.

If then you type the command:

DCS:100 115[Retrn]

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

Purpose Enters byte values into memory at the specified <address>.

Syntax E<address>[<list>][Retrn]

Comments If the optional <list> of values is typed, the replacement of byte values occurs automatically. (If an error occurs, no byte values are changed.)

If the <address> is typed without the optional <list>, DEBUG displays the address and its contents, then repeats the address on the next line and waits for your input. At this point,the Enter command waits for you to perform one of the following actions:

- ☆ Replace a byte value with a value you type. Simply type the value after the current value. If the value typed in is not a legal hexadecimal value or if more than two digits are typed, the illegal or extra character is not echoed.
- ☆ Press the space bar to advance to the next byte. To change the value, simply type the new value as described above. If you space beyond an 8-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.
- ☆ Type a hyphen (-) to return to the preceding byte. If you decide to change a byte behind the current position, typing the hyphen returns the current position to the previous byte. When the hyphen is typed, a new line is started with the address and its byte value displayed.
- ☆ Press [Retrn] to terminate the Enter command. [Retrn] may be pressed at any byte position.

Example Assume that the following command is typed

ECS:100[Retrn]

DEBUG displays.

04BA:0100 EB.__

To change this value to 41, type 41 as shown:

04BA.0100 EB.**41**__

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

04BA:0100	EB 41	10.	00.	BC __

To change BC to 42:

04BA:0100	EB.41	10.	00.	BC.**42**__

Now, realizing that 10 should be 6F, type the hyphen as many times as needed to return to byte 0101 (value 10), then replace 10 with 6F:

04BA:0100	EB.41	10.	00.	BC.42-
04BA:0102	00.-__			
04BA:0101	10.**6F**__			

Pressing [Retrn] ends the Enter command and returns to the DEBUG command level.

Purpose Fills the addresses in the <range> with the values in the <list>.

Syntax F<range> <list>[Retrn]

Comments If the <range> contains more bytes than the number of values in the <list>, the <list> will be used repeatedly until all bytes in the <range> are filled. If the <list> contains more values than the number of bytes in the <range>, the extra values in the <list> will be ignored. If any of the memory in the <range> is not valid (bad or nonexistent), an error will occur in all succeeding locations.

Example Assume that the following command is typed:

F04BA:100 L 100 42 45 52 54 41[Retrn]

DEBUG fills memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values are repeated until all 100H bytes are filled.

Purpose Executes the program currently in memory

Syntax G[=<address>[<address>]][Retrn]

Comments If only the Go command is typed, the program executes as if the program had run outside DEBUG

If =<address> is set, execution begins at the address specified The equal sign (=) is required, so that DEBUG can distinguish the start =<address> from the breakpoint <address>es

With the other optional addresses set, execution stops at the first <address> encountered, regardless of that address' position in the list of addresses to halt execution or program branching When program execution reaches a breakpoint, the registers, flags, and decoded instruction are displayed for the last instruction executed (The result is the same as if you had typed the Register command for the breakpoint address)

Up to ten breakpoints may be set Breakpoints may be set only at addresses containing the first byte of an 8086 opcode If more than ten breakpoints are set, DEBUG returns the BP Error message.

The user stack pointer must be valid and have 6 bytes available for this command The G command uses an IRET instruction to cause a jump to the program under test The user stack pointer is set, and the user flags, Code Segment register, and Instruction Pointer are pushed on the user stack (Thus, if the user stack is not valid or is too small, the operating system may crash) An interrupt code (0CCH) is placed at the specified breakpoint address(es)

When an instruction with the breakpoint code is encountered, all breakpoint addresses are restored to their original instructions. If execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions

Example Assume that the following command is typed:

GCS:7550[Retrn]

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

After a breakpoint has been encountered, if you type the Go command again, then the program executes just as if you had typed the filename at the DOS command level. The only difference is that program execution begins at the instruction after the breakpoint rather than at the usual start address.

Purpose Performs hexadecimal arithmetic on the two parameters specified.

Syntax H<value> <value>[Retrn]

Comments First, DEBUG adds the two parameters, then subtracts the second parameter from the first. The results of the arithmetic are displayed on one line; first the sum, then the difference.

Example Assume that the following command is typed:

H19F 10A[Retrn]

DEBUG performs the calculations and then displays the result:

```
02A9      0095
```

Purpose Inputs and displays one byte from the port specified by <value>.

Syntax I<value>[Retrn]

Comments A 16-bit port address is allowed.

Example Assume that you type the following command:

I2F8[Retrn]

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

42

Purpose Loads a file into memory.

Syntax L[<address> [<drive> <record> <record>]]

Comments Set BX.CX to the number of bytes read. The file must have been named either when DEBUG was started or with the N command. Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the L command is typed without any parameters, DEBUG loads the file into memory beginning at address CS:100 and sets BX:CX to the number of bytes loaded. If the L command is typed with an address parameter, loading begins at the memory <address> specified If L is typed with all parameters, absolute disk sectors are loaded, not a file The <record>s are taken from the <drive> specified (the drive designation is numeric here-- 0=A·, 1=B., 2=C , etc), DEBUG begins loading with the first <record> specified, and continues until the number of sectors specified in the second <record> have been loaded.

Example Assume that the following commands are typed·

DEBUG[Retrn]
-NFILE.COM[Retrn]

Now, to load FILE COM, type:

L[Retrn]

DEBUG loads the file and then displays the DEBUG prompt Assume that you want to load only portions of a file or certain records from a disk. To do this, type

L04BA:100 2 0F 6D[Retrn]

Example
(cont)

DEBUG then loads 109 (6D hex) records from drive C, beginning with logical record number 15 into memory beginning at address 04BA:0100. When the records have been loaded, DEBUG simply returns the - prompt.

If the file has a EXE extension, it is relocated to the load address specified in the header of the .EXE file· the <address> parameter is always ignored for .EXE files The header itself is stripped off the .EXE file before it is loaded into memory. Thus the size of an EXE file on disk will differ from its size in memory

If the file named by the Name command or specified when DEBUG is started is a .HEX file, then typing the L command with no parameters causes DEBUG to load the file beginning at the address specified in the .HEX file. If the L command includes the option <address>, DEBUG adds the <address> specified in the L command to the address found in the .HEX file to determine the start address for loading the file

Purpose Moves the block of memory specified by <range> to the location beginning at the <address> specified.

Syntax M<range> <address>[Retrn]

Comments Overlapping moves (i.e., moves where part of the block overlaps some of the current addresses) are always performed without loss of data. Addresses that could be overwritten are moved first The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block's lowest address and then to work towards the highest. The sequence for moves from lower addresses to higher addresses is to move the data beginning at the block's highest address and to work towards the lowest.

Note: If the addresses in the block being moved will not have new data written to them, the data there before the move will remain. The M command copies the data from one area into another, in the sequence described, and writes over the new addresses. This is why the sequence of the move is important.

Example Assume that you type:

MCS:100 110 CS:500[Retrn]

DEBUG first moves address CS:110 to address CS:510, then CS:10F to CS:50F, and so on until CS:100 is moved to CS:500. You should type the D command, using the <address> typed for the M command, to review the results of the move.

Purpose Sets filenames.

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

Comments The Name command performs two functions. First, Name is used to assign a filename for a later Load or Write command. Thus, if you start DEBUG without naming any file to be debugged, then the N <filename> command must be typed before a file can be loaded. Second, Name is used to assign filename parameters to the file being debugged. In this case, Name accepts a list of parameters that are used by the file being debugged.

These two functions overlap. Consider the following set of DEBUG commands:

```
-NFILE1.EXE
-L
-G
```

The results will be:

(N)ame assigns the filename FILE1.EXE to the filename to be used in any later Load or Write commands.

(N)ame also assigns the filename FILE1.EXE to the first filename parameter used by any program that is later debugged.

(N)ame sets up a File Control Block at CS:5C for FILE1.EXE.

(L)oad loads FILE1.EXE into memory.

(G)o causes FILE1.EXE to be executed with FILE1.EXE as the single filename parameter (that is, FILE1.EXE is executed as if FILE1.EXE had been typed at the command level).

Chain of Commands

A more useful chain of commands might look like this:

```
-NFILE1.EXE
-L
-NFILE2.DAT FILE3.DAT
-G
```

Here, Name sets FILE1.EXE as the filename for the subsequent Load command. The Load command loads FILE1.EXE into memory, and then the Name command is used again, this time to specify the parameters to be used by FILE1.EXE. Finally, when the Go command is executed, FILE1.EXE is executed as if FILE1 FILE2.DAT FILE3.DAT had been typed at the DOS command level.

Note: If a Write command were executed at this point, then FILE1.EXE--the file being debugged--would be saved with the name FILE2.DAT. To avoid such undesired results, you should always execute a Name command before either a Load or a Write.

There are four regions of memory that can be affected by the Name command:

CS:5C FCB for file 1
CS:6C FCB for file 2
CS:80 Count of characters
CS:81 All characters typed

Chain of Commands (cont.)

A File Control Block (FCB) for the first filename parameter given to the Name command is set up at CS:5C. If a second filename parameter is typed, then an FCB is set up for it beginning at CS:6C. The number of characters typed in the Name command (exclusive of the first character, "N") is given at location CS:80. The actual stream of characters given by the Name command (again, exclusive of the letter "N") begins at CS:81.

Note: This stream of characters may contain switches and delimiters that would be legal in any command typed at the DOS command level.

Example

A typical use of the Name command is:

DEBUG PROG.COM[Retrn]
-NPARAM1 PARAM2 /C[Retrn]
-G[Retrn]
-

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

PROG PARAM1 PARAM2 /C[Retrn]

Testing and debugging therefore reflect a normal runtime environment for PROG.COM.

Purpose Sends the <byte> specified to the output port specified by <value>.

Syntax O<value> <byte>[Retrn]

Comments A 16-bit port address is allowed.

Example Type:

O2F8 4F[Retrn]

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

Purpose Terminates the DEBUG utility.

Syntax Q[Retrn]

Comments The Q command takes no parameters and exits DEBUG without saving the file currently being operated on. You are returned to the DOS command level.

Example To end the debugging session, type:

Q[Retrn]

DEBUG has been terminated, and control returns to the DOS command level.

Purpose Displays the contents of one or more CPU registers.

Syntax R[<register-name>][Retrn]

Comments If no <register-name> is typed, the R command dumps the register save area and displays the contents of all registers and flags.

If a register name is typed, the 16-bit value of that register is displayed in hexadecimal, and then a colon appears as a prompt. You then either type a <value> to change the register, or simply press **[Retrn]** 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, type the opposite two-letter code. The flags are either set or cleared.

The flags and their codes for SET and CLEAR are listed in the table on the next page.

Comments
(cont.)

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

Whenever you type the command RF, the flags are displayed in the order shown above in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a hyphen (-). You may enter new flag values as alphabetic pairs. The new flag values can be entered in any order. You do not have to leave spaces between the flag entries. To exit the R command, press

[Retrn]

Flags for which new values were not entered remain unchanged.

If more than one value is entered for a flag, DEBUG returns a DF Error message. If you enter a flag code other than those shown above, DEBUG returns a BF Error message. In both cases, the flags up to the error in the list are changed; flags at and after the error are not.

At startup, the segment registers are set to the bottom of free memory, the Instruction Pointer is set to 0100H, all flags are cleared, and the remaining registers are set to zero.

Example

Type:

R[Retrn]

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=O4BA
IP=011A NV UP DI NG NZ AC PE NC
04BA:011A CD21 INT 21

If you type:

RF[Retrn]

DEBUG will display the flags:

NV UP DI NG NZ AC PE NC - __

Now, type any valid flag designation, in any order, with or without spaces.

For example:

NV UP DI NG NZ AC PE NC - **PLEICY[Retrn]**

DEBUG responds only with the DEBUG prompt. To see the changes, type either the R or RF command:

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

Press

[Retrn]

to leave the flags this way, or specify different flag values

Purpose Searches the <range> specified for the <list> of bytes specified.

Syntax S<range> <list>[Retrn]

Comments The <list> may contain one or more bytes, each separated by a space or comma. If the <list> contains more than one byte, only the first address of the byte string is returned If the <list> contains only one byte, all addresses of the byte in the <range> are displayed.

Example If you type:

SCS:100 110 41[Retrn]

DEBUG will display a response similar to this·

```
04BA:0104
04BA:010D
-
```

DEBUG COMMAND
(T)race Command

Purpose Executes one instruction and displays the contents of all registers and flags, and the decoded instruction.

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

Comments If the optional =<address> is typed, tracing occurs at the =<address> specified. The optional <value> causes DEBUG to execute and trace the number of steps specified by <value>.

The T command uses the hardware trace mode of the 8086 or 8088 microprocessor. Consequently, you may also trace instructions stored in ROM (Read Only Memory).

Example Type:

T[Retrn]

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=O4BA
IP=011A    NV UP DI NG NZ AC PE NC
04BA:011A     CD21                INT      21
```

If you type:

T=011A 10[Retrn]

DEBUG executes sixteen (10 hex) instructions beginning at 011A in the current segment, andthen 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 [Ctrol]S suspends the display at any point, so that you can study the registers and flags for any instruction.

Purpose Disassembles bytes and displays the source statements that correspond to them, with addresses and byte values.

Syntax U[<range>][Retrn]

Comments The display of disassembled code looks like a listing for an assembled file. If you type the U command without parameters, 20 hexadecimal bytes are disassembled at the first address after that displayed by the previous Unassemble command. If you type the U command with the <range> parameter, then DEBUG disassembles all bytes in the range If the <range> is given as an <address> only, then 20H bytes are disassembled instead of 80H.

Example Type:

U04BA:100 L10[Retrn]

DEBUG disassembles 16 bytes beginning at address 04BA:0100:

```
04BA:0100   206472   AND   [SI+72],AH
04BA:0103   69       DB    69
04BA:0104   7665     JBE   016B
04BA:0106   207370   AND   [BP+DI+70],DH
04BA:0109   65       DB    65
04BA:010A   63       DB    63
04BA:010B   69       DB    69
04BA:010C   66       DB    66
04BA 010D   69       DB    69
04BA.010E   63       DB    63
04BA:010F   61       DB    61
```

Example (cont.)

If you type

U04ba:0100 0108[Retrn]

the display will show:

```
04BA.0100    206472    AND    [SI+72],AH
04BA:0103    69        DB     69
04BA·0104    7665      JBE    016B
04BA:0106    207370    AND    [BP+DI+70],DH
```

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

Purpose Writes the file being debugged to a disk file.

Syntax W[<address>[<drive> <record> <record>]][Retrn]

Comments If you type W with no parameters, BX:CX must already be set to the number of bytes to be written; the file is written beginning from CS:100. If the W command is typed with just an address, then the file is written beginning at that address. If a G or T command has been used, BX:CX must be reset before using the Write command without parameters.

Note: If a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file (as long as the length has not changed).

The file must have been named either with the DEBUG invocation command or with the N command (refer to the Name command earlier in this manual) or you will get the message "File creation error" when you try to write the file. Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the W command is typed with parameters, the write begins from the memory address specified; the file is written to the <drive> specified (the drive designation is numeric here--0=A, 1=B, 2=C, etc.); DEBUG writes the file beginning at the logical record number specified by the first <record>, DEBUG continues to write the file until the number of sectors specified in the second <record> have been written.

★ ★ CAUTION ★ ★

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

Example Type.

W[Retrn]

DEBUG will write the file to disk, display the number of bytes written and then display the DEBUG prompt. The result might be.

Writing 4A0F bytes

-

If you type.

WCS:100 1 37 2B[Retrn]

DEBUG writes out the contents of memory, beginning with the address CS:100 to the disk in drive B. The data written out starts in disk logical record number 37H and consists of 2BH records. When the write is complete, DEBUG displays the number of bytes written, and then displays the DEBUG prompt:

Writing 1580 bytes
-

Error Message Display

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

Error Code	Explanation	Cause
BF	Bad flag	You attempted to alter a flag, but the characters typed were not one of the acceptable pairs of flag values. See the Register command for the list of acceptable flag entries.
BP	Too many breakpoints	You specified more than ten breakpoints as parameters to the G command. Retype the Go command with ten or fewer breakpoints.
BR	Bad register	You typed the R command with an invalid register name. See the Register command for the list of valid register names.
DF	Double flag	You typed two values for one flag. You may specify a flag value only once per RF command.

APPENDIX A

USING SIMULATED SECOND DISKETTE DRIVE

Introduction

Even though this system only contains one diskette drive, you can perform operations with DOS commands that require two diskettes. This is possible through the simulated second diskette drive, or drive B.

The diskette drive is usually called drive A When an operation requires two diskette drives, the diskette drive can be called drive A or drive B, depending on whether the system is seeking the first or the second diskette in the operation

Copying Files

If you specify drive B when the "A" diskette was last used, you are prompted to insert the diskette for drive B. For example, suppose you want to copy a file from one actual diskette to another After you have requested the diskette drive, insert the diskette containing the file (the source diskette) in the drive and type:

COPY TEXT.DOC B:[Retrn]

DOS reads the file from the diskette and prompts you with.

Insert diskette for drive B
and strike any key when ready

Remove the "A" diskette and replace it with the "B" (destination) diskette and press any key. DOS copies the file, then displays:

1 File(s) copied
A>

If you specify drive A when the "B" diskette was last used, you are prompted again to change diskettes This time, DOS prompts you to insert the "A" diskette.

The same procedure is used if a command is executed from a batch file. DOS prompts you to insert the appropriate diskette and press any key before it continues

Default Diskette Drive

The letter displayed in the system prompt represents the default drive, not the last diskette used. For example, assume that A is the default drive. If you had previously typed DIR B, DOS believes the "B" diskette is still in the drive. However, the prompt is still A>, because A is still the default drive. If you then type DIR with no drive letter, DOS prompts you for the "A" diskette because drive A is the default drive, and you did not specify another drive with the DIR command.

APPENDIX B

DISK ERRORS

Introduction

If a disk or device error occurs at any time during a command or program, DOS returns an error message in the following format:

<yyy> ERROR WHILE <I/O action> ON DRIVE x
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 <x> indicates the drive in which the error has occurred.

Responses

DOS waits for you to enter one of the following responses.

A *Abort*. Terminate the program requesting the disk read or write.

I *Ignore* Ignore the bad sector and pretend the error did not occur.

R *Retry*. Repeat the operation. This response is to be used when the operator has corrected the error (such as with NOT READY or WRITE PROTECT errors)

Responses 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 drive or diskette)

One other error message might be related to faulty disk read or write.

FILE ALLOCATION TABLE BAD FOR DRIVE x

This message means that the copy in memory of one of the allocation tables has pointers to nonexistent blocks. Possibly the drive or diskette was incorrectly formatted or not formatted before use. If this error persists, the drive or diskette is currently unusable and must be formatted prior to use.

APPENDIX C

ANSI ESCAPE SEQUENCES

Introduction An ANSI escape sequence is a series of characters (beginning with an escape character or keystroke) that you can use to define functions to DOS. Specifically, you can reassign keys, change graphics functions, and affect cursor movement

This appendix explains how the ANSI escape sequences are defined. Examples on how to use ANSI escape sequences are included at the end of this appendix.

Note: The default value is used when no explicit value or a value of zero is specified.

Pn represents "numeric parameter." This is a decimal number specified with ASCII digits.

Ps represents "selective parameter " This is any decimal number that is used to select a subfunction. Multiple subfunctions may be selected by separating the parameters with semicolons.

Cursor Functions

The following escape sequences affect the cursor position on the screen:

CUP - Cursor Position

ESC [Pl ; Pc H

HVP - Horizontal & Vertical Position

ESC [Pl ; Pc f

CUP and HVP move the cursor to the position specified by the parameters. The first parameter specifies the line number, and the second parameter specifies the column number. The default value is 1. When no parameters are specified, the cursor is moved to the home position

CUU - Cursor Up

ESC [Pn A

This sequence moves the cursor up one line without changing columns. The value of Pn determines the number of lines moved. The default value for Pn is 1 The CUU sequence is ignored if the cursor is already on the top line.

CUD - Cursor Down

ESC [Pn B

This sequence moves the cursor down one line without changing columns. The value of Pn determines the number of lines moved The default value for Pn is 1. The CUD sequence is ignored if the cursor is already on the bottom line

CUF - Cursor Forward

ESC [Pn C

The CUF sequence moves the cursor forward one column without changing lines The value of Pn determines the number of columns moved. The default value for Pn is 1. The CUF sequence is ignored if the cursor is already in the far right column

Cursor Functions (cont.)

CUB - Cursor Backward

ESC [Pn D

This escape sequence moves the cursor back one column without changing lines. The value of Pn determines the number of columns moved. The default value for Pn is 1. The CUB sequence is ignored if the cursor is already in the far left column.

DSR - Device Status Report

ESC [6 n

The console driver will output a CPR sequence (see below) on receipt of the DSR escape sequence.

CPR - Cursor Position Report (from console driver to system)

ESC [Pn ; Pn R

The CPR sequence reports current cursor position via standard input. The first parameter specifies the current line and the second parameter specifies the current column.

SCP - Save Cursor Position

ESC [s

The current cursor position is saved. This cursor position can be restored with the RCP sequence (see below).

RCP - Restore Cursor Position

ESC [u

This sequence restores the cursor position to the value it had when the console driver received the SCP sequence.

Erasing

The following escape sequences affect erase functions.

ED - Erase Display

ESC [2 J

The ED sequence erases the screen, and the cursor goes to the home position.

EL - Erase Line

ESC [K

This sequence erases from the cursor to the end of the line (including the cursor position).

Modes of Operation

The following escape sequences affect screen graphics

SGR - Set Graphics Rendition

ESC [Ps , . , Ps m

The SGR escape sequence invokes the graphic functions specified by the parameter(s) described below. The graphic functions remain until the next occurrence of an SGR escape sequence.

SGR Escape Sequence

Parameter	Parameter Function	
0	All attributes off	
1	Bold on	
4	Underscore on	(monochrone displays only)
5	Blink on	
7	Reverse Video on	
8	Concealed on	(ISO 6429 standard)
30	Black foreground	(ISO 6429 standard)
31	Red foreground	(ISO 6429 standard)
32	Green foreground	(ISO 6429 standard)
33	Yellow foregound	(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)

Modes of Operation

SM - Set Mode

ESC [= Ps h
or ESC [= h
or ESC [= 0 h
or ESC [? 7 h

The SM escape sequence changes the screen width or type to one of the following parameters:

Parameter	Parameter Function
0	40 x 25 black and white
1	40 x 25 color
2	80 x 25 black and white
3	80 x 25 color
4	320 x 200 color
5	320 x 200 black and white
6	640 x 200 black and white
7	wrap at end of line

RM - Reset Mode

ESC [= Ps l
or ESC [= l
or ESC [= 0 l
or ESC [? 7 l

Parameters for RM are the same as for SM (Set Mode), except that parameter 7 will reset the wrap at the end of line mode.

Keyboard Reassignment

Although not part of the ANSI 3.64-1979 or ISO 6429 standard, the following keyboard reassignments are compatible with these standards

The control sequence is.

```
        ESC [ Pn, Pn;  .  Pn p
or      ESC [ ''string'', p
or      ESC [ Pn, ''string'', Pn, Pn, ''string'', Pn p
or      any other combination of strings and decimal
        numbers
```

The final code in the control sequence (p) is one reserved for private use by the ANSI 3.64-1979 standard

The first ASCII code in the control sequence defines which code is being mapped The remaining numbers define the sequence of ASCII codes generated when this key is intercepted. 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 For example:

Reassign the Q and q key to the A and a key (and vice versa).

ESC [6 5 , 8 1 p	A becomes Q
ESC [9 7 , 1 1 3 p	a becomes q
ESC [8 1 , 6 5 p	Q becomes A
ESC [1 1 3 , 9 7 p	q becomes a

Reassign the F10 key to to a DIR command followed by a carriage return

ESC [0 , 6 8 , '' d i r '' , 1 3 p

The 0,68 is the extended ASCII code for the F10 key, 13 decimal is a carriage return

APPENDIX D

HOW TO CONFIGURE YOUR SYSTEM

Introduction The DOS configuration file (CONFIG.SYS) allows you to add device drivers which will be activated for you when you sign on. The configuration file is simply an ASCII file that has certain commands for DOS start up. The start up (boot) process is as follows:

1. The disk boot sector is read. This contains enough code to read DOS code and the installation's BIOS (machine-dependent code).
2. The DOS code and BIOS are read.
3. A variety of BIOS initializations are done.
4. A system initialization routine reads the configuration file (CONFIG.SYS), if it exists, to perform device installation and other user options. Its final task is to execute the command interpreter, which finishes the DOS boot process.

Note: Any CONFIG.SYS file on drive D will be executed only if the AUTOEXEC.BAT file is present on drive D. Refer to AUTOEXEC.BAT for more information on this topic.

Method

The CONFIG.SYS file is located on drive C. Copy it from drive C into the root directory of your drive D. Then choose the commands to add or change from the list below. The default values already in the file are also listed.

Commands

BUFFERS = <number>
This is the number of sector buffers that DOS will use for disk buffering. The default is 2.

FILES = <number>
This is the maximum number of files that DOS allows to be open. If this command isn't specified in the CONFIG.SYS file, then the default is 8.

DEVICE = <filename>
This installs the device driver in <filename> into the system list. (See below.)

BREAK = <ON or OFF>
If ON is specified (the default is OFF), a check for [Ctrl]C as input will be made every time the system is called. ON improves the ability to cancel programs.

SHELL = <filename>
This begins execution of the shell (top-level command processor) from <filename>.

Example

A typical configuration file might look like this:

```
Device   = C:\SYSTEM\NSTAR1.DRV
Device   = C:\SYSTEM\ANSI.SYS
Files    = 12
Buffers  = 10
```

APPENDIX E

TROUBLESHOOTING GUIDE

Symptom	Solution
The CAPS LOCK and/or NUM LOCK keys do not indicate the correct state	Turn off the lights and sign off and then sign on again.
A workstation does not respond to the keyboard	First be certain that there is a problem. Certain tasks may take a long time for the workstation to respond. If there is a problem, use the RESTART command from another workstation.
Cannot delete certain files or cannot overwrite them	Certain files created by DOS, such as COMMAND.COM and the files in subdirectory SYSTEM are read-only. You should not attempt to alter them.
Unable to boot from a diskette	There is a special procedure for booting the diskette on the DIMENSION. See the System Manual for instructions.

INDEX

ANSI Escape Sequence, C-1
Arguments, 3-6
ASSIGN command, 4-8
AUTOEXEC.BAT file, 3-11
Automatic program execution, 3-11
AUX extension, 1-11
BACKUP Command, 4-11
BACKUP and RESTORE, 3-22
Backing up files, 1-13, 3-22
    see also COPY command
Batch processing, 3-9
    automatic, 3-11
    cancelling, 3-10
    commands, 4-7
    files, 3-9
    parameters, 3-14, 4-91
    suspending, 4-90
BREAK command, 4-15
Cancelling commands, 3-7
CD, *see* CHDIR command
Changing directories, 2-8, 4-16
Changing filenames, 4-65
Changing the date, 4-29
Changing the time, 4-77
CHDIR command, 2-8, 4-16
Checking the date, 4-29
CHKDSK command, 4-17
Clock, 4-77
CLS command, 4-21
Command processor, *see* COMMAND.COM
COMMAND.COM, 1-3, 3-3
Commands, 3-2
    arguments, 3-6
    ASSIGN, 4-8
    batch processing, 3-9
    BREAK, 4-15
    cancelling, 3-7

Commands (cont.)
CHDIR, 2-8, 4-16
CHKDSK, 4-17
CLS, 4-21
COMMAND, 4-22
COPY, 1-12, 4-23
CTTY, 4-27
CURSOR, 4-28
DATE, 4-29
DEL, 4-30
delimiters, 3-7
DIR, 1-6, 4-31
DISKBOOT, 4-32
DISKCOMP, 4-33
DISKCOPY, 4-34
DISKLOCK, 4-36
ECHO, 4-86
EXE2BIN, 4-37
EXIT, 4-40
external, 3-3
filters, 3-19
FIND, 3-19, 4-41
FOR, 4-87
format, 4-2, 4-43
GOTO, 4-88
IF, 4-89
internal, 3-3
keys to use, 3-7, 5-4
list, 4-3
LOCAL, 4-45
MKDIR, 2-12, 4-49
MORE, 3-19, 4-50
optional, 3-5
options, 3-5
PATH, 2-10, 4-51
PAUSE, 3-9, 4-90
piping, 3-20
PRINT, 4-53

Commands (cont.)
PRINTER, 4-56
PROFILES, 4-59
PROMPT, 4-60
RECOVER, 4-62
RELEASE, 4-63
REM, 3-9, 4-64
REN, 4-65
REQUEST, 4-66
RESTART, 4-67
RMDIR, 4-72
SET, 4-73
SHIFT, 4-91
SIGNOFF, 4-74
SORT, 3-19, 4-75
STATS, 4-76
switches, 3-6
synonyms, 4-3
template, 5-2
TYPE, 4-80
typing, 3-7, 5-2
USERS, 4-81
VER, 4-82
VERIFY, 4-83
VOL, 4-84
Comparing files, 7-1
CON extension, 1-11
Concatenation, 4-25
CONFIG.SYS files, D-1
commands, D-2
Configuration, D-1
Configuring workstation devices, 4-45
COPY command, 1-12, 4-23
Copying files, 1-12, 4-23
Creating directories, 2-12
Creating files, 1-4, 6-1
CTTY command, 4-27
Cursor control, 5-7

DATE command, 4-29
DEBUG, 9-1
    .EXE files, 9-23
    cancelling, 9-1
    errors, 9-39
    flags, 9-31
    parameters, 9-6, 9-20
    starting, 9-2
DEBUG commands, 9-5
    Assemble, 9-10
    Compare, 9-12
    Dump, 9-13
    Enter, 9-15
    Fill, 9-17
    Go, 9-18
    Hex, 9-20
    Input, 9-21
    Load, 9-22
    Move, 9-24
    Name, 9-25
    Output, 9-28
    Quit, 9-29
    Register, 9-30
    Search, 9-33
    Trace, 9-34
    Unassemble, 9-35
    Write, 9-37
Default drive, 1-3
Default printer, 3-12
DEL command, 4-30
Deleting files, 4-30
Destination drive, 3-8
Device drivers, C-1
Device names, 1-11
DIR command, 1-6, 4-31
Directories, 1-5, 2-2
    creating, 2-12
    error checking, 4-17

Directories (cont.)
    examples, 2-3
    hierarchical structure, 2-2
    naming, 2-12
    parent, 2-10
    removing, 4-72
    root, 1-5, 2-2
    shorthand notations, 2-10
    working, 1-5, 2-11
Disk errors, B-1
DISKBOOT, 4-32
DISKCOMP, 4-33
DISKCOPY command, 4-34
Diskette drive, 1-2, A-1
    releasing, 4-63
    requesting, 4-66
Diskettes, 1-2
    copying, 4-34
    formatting, 4-43
    recovering bad, 4-62
    using two, A-1
Displaying file contents, 4-80
Displaying files, 1-6
Displaying signed-on users, 4-81
DOS, 4
    commands, 3-2
    memory, 8-3
    prompt, 1-3
    starting up, 1-3
    version number, 4-82
Drives, 1-2
    default, 1-3
    destination, 3-8
    formatting, 4-43
    letters, 1-2, 1-7
    reassigning, 4-8
    source, 3-8
    space, 4-76

ECHO command, 4-86
EDLIN, 6-1
    creating files, 6-3
    editing files, 6-3
    editing keys, 6-5
    errors,, 6-47
    list of commands, 6-17
    saving files, 6-4
EDLIN commands, 6-15
    Append, 6-19
    Copy, 6-20
    Delete, 6-23
    Edit, 6-25
    End, 6-27
    Insert, 6-28
    List, 6-32
    Move, 6-35
    options, 6-17
    Page, 6-36
    Quit, 6-37
    Replace, 6-38
    Search, 6-42
    Transfer, 6-45
    typing, 6-15
    Write, 6-46
ENHANCE command, 4-36
ERASE, *see* DEL
Errors,
    DEBUG, 9-39
    directories, 4-17
    disk, B-1
    FC, 7-12
    MS-LINK, 8-24
EXE2BIN command, 4-37
EXIT command, 4-40
Extensions, 1-7
External commands, 3-3
FC, 7-1
    errors, 7-12
    examples, 7-8

FC (cont.)
    limitations, 7-2
    redirecting output, 7-7
    reporting, 7-6
    switches, 7-4
    syntax, 7-3
File allocation table, 1-5
File comparison utility, *see* FC
Filenames, 1-7
    extension, 1-7
    reserved, 1-11
    wild cards, 1-9
Files, 1-4
    backup, 1-13
    binary format, 4-37
    comparing, 7-1
    concatenation, 4-25
    copying, 1-12, 4-23
    creating, 1-4, 6-1
    displaying, 1-6
    naming, 1-7
    organizing, 2-2
    pathnames, 2-6
    paths, 2-6
    printing, 4-53
    protecting, 1-13
    renaming, 4-65
    specification, 1-8
Filespec, 1-8
Filters, 3-19
FIND command, 3-19, 4-41
Fixed disks, 4-76
FOR command, 4-87
FORMAT command, 4-43
Function keys, 5-4, 6-5
GOTO command, 4-88
IF command, 4-89
Input redirection, 3-17

Internal commands, 3-3
Keys, 5-3
    cursor control keys, 5-7
    function keys, 5-4, 6-5
    re-assigning, C-1
Line editor, 6-1
Linking programs, 8-1
LOCAL command, 4-45
Local printers, 4-45
Locating files, 2-6
LST extension, 1-11
MD, *see* MKDIR
Memory, 8-3
MKDIR command, 2-12, 4-49
Modems, 4-45
MORE command, 3-19, 4-50
Mouse, 4-45
MS-DOS, 4
MS-LINK, 8-1
    cancelling, 8-13
    command characters, 8-12
    errors, 8-24
    files, 8-5
    memory, 8-3
    prompts, 8-14
    sample sessions, 8-21
    starting, 8-7
    switches, 8-17
Naming directories, 2-12
Naming files, 1-7
NUL extension, 1-11
Operating system, 4
Options
    commands, 3-5
    EDLIN, 6-17
Organizing files, 2-2
Output redirection, 3-18

Parameters
batch processing, 3-14, 4-91
DEBUG, 9-6, 9-20
Parent directory, 2-10
PATH command, 2-10, 4-51
Pathnames, 2-6
Paths, 2-6
examples, 2-7
PAUSE command, 3-9, 4-90
Piping commands, 3-20
PRINT command, 4-53
Print lists, 4-53
PRINTER command, 4-56
Printers, 4-56
Printing files, 4-53
PRN extension, 1-11
PROFILES command, 4-59
Programs
debugging, 9-1
linking, 8-1
Prompt, 1-3
changing, 4-60
PROMPT command, 4-60
Protecting files, 1-13
RD, *see* RMDIR
Reassigning drives, 4-8
RECOVER command, 4-62
RELEASE command, 4-63
REM command, 3-9, 4-64
Removing directories, 4-72
REN command, 4-65
RENAME, *see* REN
Renaming files, 4-65
REQUEST command, 4-66
Reserved filenames, 1-11
RESTART command, 4-67
Restarting a workstation, 4-67
RESTORE command, 4-68

RMDIR command, 4-72
Root directory, 1-5, 2-2
Searching files, 4-41
SET command, 4-73
SHIFT command, 4-91
Sign on, 1-3, 3-11
Signing off, 4-74
SIGNOFF command, 4-74
Simulated second diskette drive, A-1
SORT command, 3-19, 4-75
Sorting data, 4-75
Source drive, 3-8
STATS command, 4-76
Stopping the display, 3-8
Subdirectories, 2-2
Switches, 3-6
Synonyms, 4-3
Template, 5-2
Testing programs, 9-1
TIME command, 4-77
TIMER command, 4-92
TREE command, 4-78
TYPE command, 4-80
Typing commands, 3-7, 5-2
USERS command, 4-81
VER command, 4-82
VERIFY command, 4-83
VOL command, 4-84
Volume label, 4-84
Wild cards, 1-9
    *, 1-9
    ?, 1-9
    examples, 1-10
Working directory, 1-5
    changing, 4-16
    displaying, 2-11
Workstation devices, 4-45

Reader Comments

Please use this postpaid mailer to make us aware of the strengths and weaknesses of this North Star manual. Specific errors or deficiencies should be referenced by page numbers and paragraph headings. Attach additional sheets if you require more space for your comments

We will carefully consider your suggestions for incorporation in future versions of this manual Thank you

Do you find the manual easy to use and understand? ☐ YES ☐ NO

Do you think certain aspects should be organized differently? If so, please explain below

Are there specific points or issues in the manual that need clarification or correction? Please give reference page numbers and paragraph/table/figure headings and describe the problem below

Additional comments

NAME OF MANUAL _____ PART NUMBER _____

NAME _____ DATE _____

COMPANY NAME _____

STREET ADDRESS _____

CITY _____ STATE _____ ZIP CODE _____

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 2106 SAN LEANDRO, CA

POSTAGE WILL BE PAID BY ADDRESSEE

North Star Computers, Inc.
ATTENTION: Technical Publications
14440 Catalina Street
San Leandro, CA 94577

NOTICE

DIMENSION Operating System
DOS Revision 1.2.0

Dear North Star Customer,

This revision of the North Star DIMENSION Operating System, DOS 1.2.0, contains a variety of enhancements over previous releases. These enhancements are listed at the end of this notice.

You will need the documentation for DOS 1.1.0 as a base for updating to 1 2.0. The documentation included with the 1.2.0 Operating System revision consists of:

- ☆ DIMENSION System Manual Addendum, part number 04114A, which brings the system manual up to revision level 03550C
- ☆ DIMENSION Operating System Manual Addendum, part number 04188A, which brings the DOS manual up to revision level 03589D

You will also need

- ☆ DIMENSION User's Guide, 03551B **or**
- ☆ DIMENSION User's Guide, part number 03551A plus Addendum 04133A

Please be sure you are using the most current documentation. If you do not already have DOS 1.1.0 documentation, see your North Star DIMENSION dealer.

If you already have DOS 1.0.0, 1.0.2 or 1.1.0 installed on your system and you will be using the enclosed diskettes to update your operating system to 1.2.0, follow the installation procedure on the next page. **Do not use SETUP to install an operating system update.**

Before you begin the UPDATE procedure, format a diskette. This diskette will be required during the procedure to back up your system descriptor files.

DIMENSION DOS 1.2.0 Revision
Using UPDATE

Introduction Use this procedure only if you already have DOS 1.0.0, 1.0.2 or 1.1.0 installed on the fixed disk. To run properly, UPDATE requires that at least two partitions, drive C and drive D, be assigned to the manager.

Procedure: Using UPDATE to Install DOS 1.2.0

1. Sign on as the system manager and enter manager mode. Request the diskette drive by typing

   REQUEST D[Retrn]

2. Insert System Diskette #1 in the diskette drive and type

   A:UPDATE[Retrn]

   Follow the prompts on the screen.

3. A prompt asks you to select a path to which the utility routines will be copied. To select the default path (to drive C), press

   [Retrn]

   The utilities will be copied to drive C. If you want the utilities copied to another directory, or stored within a subdirectory, simply specify the desired path.

   Note: The path you select must contain the old utilities (the program looks specifically for MAINT.EXE to validate the selected path). Be sure that all of the old utilities are in the same location (on the same directory or subdirectory), so that they will be correctly updated. If any old utilities have been moved to other locations, they must be removed or replaced after UPDATE has been run. Failing to do so could create problems while running certain programs.

   UPDATE will ask for a formatted diskette to back up your system descriptor files. After UPDATE has transferred the new utilities, you can then, of course, move the individual files to any desired directory or subdirectories.

Procedure: Using UPDATE to Install DOS 1.2.0

4. When the UPDATE program is finished, remove the diskette from the drive and store the three system diskettes and the system descriptor backup diskette in a safe place.

5. UPDATE has saved your user profiles and drive assignments; however, you will need to reinstall your system printers. Follow the procedures in Chapter 4 of your DIMENSION System Manual, part number 03550 revision B or C.

6. UPDATE has installed new COMMAND.COM and CONFIG.SYS files on drive C. You must copy CONFIG.SYS from drive C to drive D for each user who has an AUTOEXEC.BAT on their drive D.

This concludes the operating system update procedure. You may want to change the type of one or more partitions, or create new partitions to make use of the shared drive capabilities. Be sure to read about using shared drives and changing partition types in your System Manual before you make any changes.

Feature Enhancements

Tape Backup is now available as a hardware option. The software is included with DOS 1.2.0.

Compatibility Enhancements

Open Access will run on DOS 1.2.0. The local asynchronous port emulation has improved significantly, and several more serial packages will run on DOS 1.2.0.

- ☆ PhoneLink
- ☆ PC Talk
- ☆ Transend

The Mouse Systems Mouse will work with AutoCAD in DOS 1 2 0

Other Enhancements

UPDATE has been modified so you can run it from any partition.

The print spooler time-out pause will default to five seconds instead of 15 seconds and the minimum value is two seconds.

The boot process after sign-on has been modified to load COMMAND.COM from drive C irrespective of the location of AUTOEXEC.BAT. Otherwise, the treatment of AUTOEXEC.BAT and CONFIG.SYS is unchanged; therefore, it is no longer necessary to have COMMAND.COM on drive D. If AUTOEXEC.BAT is on drive D, DOS will execute it. If CONFIG.SYS is *also* on drive D, DOS will execute it as well. We suggest that you delete all copies of COMMAND.COM except the one on drive C.

CONFIG.SYS File

The CONFIG.SYS file in DOS 1.2.0 causes two drivers to be loaded: NSTAR1.DRV and ANSI.SYS. These drivers are necessary for certain applications to run on the DIMENSION.

Some applications will not run in 128K workstations with these drivers installed. NSTAR1.DRV uses 10,048 bytes of workstation memory space. ANSI.SYS uses 1568 bytes If you need more memory, try deleting the lines

DEVICE=C:\SYSTEM\NSTAR1.DRV
DEVICE=C:\SYSTEM\ANSI.SYS

from your CONFIG.SYS file. Be sure your CONFIG.SYS file is in the same drive as your AUTOEXEC.BAT. And remember, you must sign off and sign on again for any changes in CONFIG.SYS to take effect.

Bug Fixes

The FORMAT command will work on partitions larger than 16 megabytes.

Several problems have been fixed in the BACKUP and RESTORE commands.

Several improvements have been made to the SETUP command. It now preserves your fixed disk #2 configuration when you re-install fixed disk #1. Also, it will not hang if you go away and leave it for a long time.

Error handling in the local asynchronous port has been modified to be more IBM compatible. The read-fault-error from CTTY (caused by typing while it's outputting) and the not-ready-error from printing (caused by having the printer set for XON/XOFF) have been eliminated.

Print Queue Limits

DOS 1.2.0 has the same print queue limits as DOS 1.1.0. These are 30 files per shared printer queue and one megabyte combined shared printer queue storage capacity If you must print files totaling more than one megabyte in size, use a local printer to avoid over-filling the shared printer queue.

DISKLOCK Command

The DOS 1.2.0 addenda move the documentation for the DISKLOCK command from the DOS Manual to the System Manual.

When running DISKLOCK on a system with only one fixed disk, the screen will not change after you press [F6]. However, the command has in fact parked the heads. If your system has two fixed disk drives, the screen will change to inform you that the heads are parked. In either case, you should turn off the central module after pressing [F6].

Notice

DIMENSION DOS 1.2.0

If you are setting up your DIMENSION system for the first time, review pages 4 through 6 of the DOS 1.2.0 Revision Notice (part number 04190) and refer to the installation instructions in your DIMENSION System Manual.

Some resident applications, such as SideKick, have been known to cause problems when running UPDATE. If you run UPDATE, be sure no resident applications are installed.

If you are installing DOS 1.2.0 on a DIMENSION system with six or more workstations, we recommend that you also install a 256K RAM expansion board. If your DIMENSION system has eleven or more workstations, the 256K RAM expansion is essential for proper operation

DOS 1.2.0 provides tape backup and restore facility in conjunction with the North Star DIMENSION Tape Backup Module. There is a limit of 900 files and subdirectories per partition for proper TAPE utility operation. If you intend to install a Tape Backup Module and you suspect any of your partitions may exceed the limit, you can run CHKDSK to obtain the number of files and subdirectories on those partitions.

The **/S** option in the RESTORE command (used in restoring data from diskettes to the fixed disk) does not work correctly on DOS 1.2.0. You may use this option together with * . * to restore an entire partition, but do not attempt to restore specific files or groups of files. If you do not use the **/S** option, RESTORE works as described in the DOS Manual.