MS-DOS GUIDE

for Attache 8:16 and Attache 8:16S

Portable Computer Systems

Published by Otrona Advanced Systems Corp.

Otrona Pub. No. 92051240 Version 1.1 November 1983

Copyright

Disclaimer

Otrona Advanced Systems Corp. makes no representations or warranties with respect to the software and documentation herein described and especially disclaims any implied warranties of merchantabilities or fitness for any particular purpose. Further, Otrona Advanced Systems Corp. reserves the right to revise this software and associated documentation and to make changes from time to time in the content without obligation of Otrona Advanced Systems Corp. to notify any person of such revisions or changes.

Trademarks

Attache 8:16 Portable Computer, Attache 8:16S Portable Computer, Valet, and Charton are trademarks of Otrona Advanced Systems Corp., Boulder, Colorado, 80301. References are made in this document to the WordStar^{R} word processing system, a registered trademark of MicroPro International Corporation, San Rafael, California, to the Control Program for Microprocessors (commonly known as CP/M), a trademark of Digital Research Inc., Pacific Grove, California, and to MS-DOS, BASIC-80 and Multiplan, trademarks of Microsoft Corporation, Bellevue, Washington.

Microsoft (R) MS(tm)-DOS User's Guide

Information in this document is subject to change without notice and does not represent a commitment on the part of the Microsoft Corporation. The software described in this document is furnished under a license agreement or non-disclosure agreement. The software may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the MS-DOS Disk Operating System on magnetic tape, disk, or any other medium for any purpose other than the purchaser's personal use.

Contents

4. Learning More About Commands

5. DOS Commands

5. DOS Commands (continued)

6. DOS Editing and Function Keys

7. EDLIN - The Line Editor

7. EDLIN - The Line Editor (continued)

8. FC - File Comparison Utility

9. Configuring Your System and Adding Drivers

9. Configuring Your System and Adding Drivers (continued)

10. Appendixes

How to USE this Guide

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

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

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

Chapter 9 tells you how to add device drivers and customize your system through the use of a configuration file named "CONFIG.SYS." This file executes automatically when MS-DOS is booted.

Appendixes to this manual include instructions for users with one-disk-drive systems, disk error messages, command reference charts, and assorted technical information.

DOS Guide

Introduction

Introduction

Overview

Microsoft MS-DOS is a disk operating system that is used with Attache 8:16 for 16-bit operation. MS is a registered trademark of Microsoft Corporation (the system author and distributor). DOS stands for Disk Operating System. You can refer to the system as either MS-DOS or DOS.

MS-DOS is a disk operating system that enables you to create and keep track of files, run and link programs, and access peripheral devices such as printers that are attached to your computer. Through MS-DOS you communicate with the computer, disk drives, and printer, managing these resources to your advantage.

The operating system is your "silent partner" when you are using the computer. It provides the interface between the hardware and you (the user) and also between the hardware and the specific application software (such as a word processing or spreadsheet program) that you may be using.

DOS Commands and Functions

In addition to internal system functions that are transparent to the casual user, MS-DOS provides a number of program commands that range from basic system and diskette maintenance functions to advanced programming and system modification functions.

These programs may be either "internal" (contained internally in the DOS command processor) or "external" (contained externally in files with a .COM or .EXE extension on the Attache DOS diskette).

Internal commands are the simplest, most commonly used commands. You cannot see these commands when you do a directory listing on your MS-DOS disk because they are part of the command processor. When you type these commands, they execute immediately.

External commands reside on disks as program files. They must be read from disk before they can execute. If the disk containing the command is not in the drive, MS-DOS will not be able to find and execute the command. When you enter an external command, do not include its filename extension.

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

Internal Commands

BREAK	Checks for a CTRL C entered at the keyboard
CHDIR	Changes directories or displays working directory (**CD**)
CLS	Clears the screen
COPY	Copies file(s) as specified
CTTY	Changes console TTY
DATE	Displays and sets date
DEL	Deletes file(s) as specified (**ERASE**)
DIR	Lists requested directory entries
ECHO	Turns batch file echo feature on or off
EXIT	Exits from secondary command processor to COMMAND.COM
FOR	Batch command for repeated command execution
GOTO	Batch command to transfer control to a specified line
IF	Batch command for conditional command execution
MKDIR	Makes a directory (**MD**)
PATH	Specifies the directories to be searched
PAUSE	Suspends execution in a batch file
PROMPT	Designates command prompt
REM	Displays a comment in a batch file
REN	Renames first file as second file (**RENAME**)
RMDIR	Removes a directory (**RD**)
SET	Sets one string value equivalent to another
SHIFT	Allows access to over 10 batch replaceable parameters
TIME	Displays and sets time
TYPE	Displays the contents of the specified file
VER	Prints MS-DOS version number
VERIFY	Turns verify switch on or off when writing to disk
VOL	Prints volume identification label

External Commands

The following external commands are described in this manual:

ARCHIVE Backs up and restores files on the hard disk (**ARK**)

CHKDSK Scans the directory of the default or designated drive and checks for consistency

CONVERT Bi-directional CP/M to MS-DOS file converter

DEBUG Program debugging tool

DISKCOPY Copies entire diskettes

EDLIN Line editor for creating source or text files

EXE2BIN Converts executable files to binary format

EXPAND Creates several lines from one list of arguments

FC Compares files

FILETYPE Changes or displays file and directory attributes

FIND Searches files for a specified text string

FIXDISK Locates bad sectors on the hard disk

FORMAT Formats a diskette for the MS-DOS operating system

HDFORMAT Formats the hard disk

LINK Combines separately produced object modules

MAKEDB Converts hex file values to decimal values

MORE Displays console output one screen at a time

MOVAFILE Transfers files with the archive attribute set

PRINT Queues and prints text files

RECOVER Recovers files from a damaged disk

SIZE Displays size of specified files or total size of files

SORT Sorts data alphabetically, forward or backward

SYS Transfers MS-DOS system files to the specified drive

TREE Displays directory paths

Syntax Notation

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

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

< > Angle brackets indicate data you must enter. When the angle brackets enclose text, 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 bar indicates an OR statement in a command. When used with an MS-DOS filter, the bar indicates a pipe.

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

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

MS-DOS Files

The Attache DOS diskette contains the following MS-DOS files:

COMMAND.COM	EXE2BIN.EXE	MAKEDB.EXE
* MSDOS.SYS	EXPAND.COM	MORE.COM
* IO.SYS	FC.EXE	MOVAFILE.COM
ARCHIVE.COM	FILETYPE.COM	PRINT.COM
CHKDSK.COM	FIND.EXE	RECOVER.COM
CONVERT.COM	FIXDISK.COM	SIZE.EXE
DEBUG.COM	FORMAT.COM	SORT.EXE
DISKCOPY.COM	HDFORMAT.COM	SYS.COM
EDLIN.COM	LINK.EXE	TREE.COM

* Indicates a "hidden" file that does not appear on the directory display.

DOS Guide

Getting Started

Getting Started

Overview

This section describes the procedures for booting the DOS operating system, formatting diskettes for use with DOS, and making backup copies of the Attache DOS diskette.

How to Boot DOS

To use Attache 8:16 with DOS, you must "load" a copy of the DOS operating system from a diskette into memory. This is called a "bootstrap" operation, or simply "booting." When DOS has been booted, you may activate applications programs that are written to perform under DOS.

To boot DOS:

1. Turn the power on. The following message is displayed:

   Otrona Attache [x]

   No Disk or Disk Not Readable
   Now in Terminal Mode

   This is the Terminal Mode prompt. DOS will be loaded from this mode.

2. Insert the Attache DOS diskette in Drive A (the upper drive) and close the drive door.

3. Press the **RESET** key and the **SHIFT** key on the right side of the keyboard at the same time to boot the system.

   Booting takes approximately 10 seconds. The message "Otrona Attache [x]" is displayed during this time. [x] identifies the ROM version for your computer.

4. Once MS-DOS has been loaded, the system searches the disk for the COMMAND.COM file (the command processor) and loads it into memory. When the command processor is loaded, the following display appears on your screen:

   Attache MS-DOS version 2.x
   Copyright 1981, 82, 83 Microsoft Corp.

   Command V. 2.02

   Current Date is DAY DATE
   Enter new date:

How to Set the Date

The displayed date is from Attache's real-time clock. If the date is correct as displayed, press **RETURN.**

If the displayed date is not correct, type today's date in the mm-dd-yy format and press **RETURN.** The real-time clock will be updated to the date you type.

The mm-dd-yy format is as follows:

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

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

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

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

6-1-82 or 06/01/82

are both acceptable answers. If you enter an invalid date or form of date, DOS prompts again to enter the new date.

How to Set the Time

When the date has been accepted, DOS prompts:

```
Current time is 8:30:00.00
Enter new time:_
```

The displayed time is from Attache's real-time clock. If the time is correct as displayed, press **RETURN.**

If the displayed time is not correct, enter the current time in the hh:mm format and press **RETURN.** The real-time clock will be updated to the time you type here.

The hh:mm format (in military time) is as follows:

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

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

MS-DOS uses this time value to keep track of when you last updated and/or created files on the system. Notice that MS-DOS uses military time. For example, 1:30 p.m. is written 13:30.

Example:

```
Current time is 0:00:14.00
Enter new time: 9:05
```

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

Note: If you make a mistake while typing, press the **CTRL** key and the **C** key at the same time. This **CTRL C** function aborts your current entry. You can then re-answer the prompt or type another command. To correct a line before you press **RETURN**, use the **BACKSPACE** key to erase one letter at a time.

You have now completed the steps for starting MS-DOS. When the time has been accepted, a prompt is displayed as shown:

```
A>
```

This is the DOS prompt, which indicates that DOS is loaded into memory and waiting for a command. The cursor is positioned to the right of the prompt.

The DOS prompt is a symbol that DOS displays when it is ready for instructions. The letter "A" indicates the logged disk drive, which is the drive currently being used for reading and writing information.

Drive A is always the logged disk drive when DOS is booted. DOS commands or applications programs can be activated when A> is displayed.

How to Change the Default Drive

You can ask MS-DOS to search the disk in Drive B by changing the drive designation or by specifying **B:** in a command. To change the disk drive designation, enter the new drive letter followed by a colon.

For example:

```
A>      (MS-DOS prompt)
A>B:    (you type B: in response to the prompt)
B>      (system responds with B> and Drive B is now the
         default or "logged" drive)
```

When the system prompt B> appears, MS-DOS searches only the disk in Drive B until you specify a different default drive.

You can change drives automatically at each cold boot by placing the command "B:" in an AUTOEXEC.BAT file, discussed in Chapter 4.

How to Format Your Disks

You must "format" all new disks before they can be used by DOS. A blank disk must be formatted with the FORMAT command. FORMAT changes the disk to a format that MS-DOS can use and also analyzes the disk for defective tracks.

Note: If the disk is not already blank, formatting it will destroy any data that exists on the disk. Do not attempt to format diskettes that contain important data or programs that you wish to keep.

The FORMAT Command

The syntax of the FORMAT command is:

FORMAT [d:] [/switches]

where d: is the drive designation (the drive that contains the disk to be formatted). Optional switches are discussed below.

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

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

FORMAT B:

MS-DOS issues the following message:

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

After you insert the new disk in Drive B and press a key, the following message is displayed:

Formatting...

The diskette is formatted as a double-sided disk at 9 sectors per track. You may override these defaults by specifying switches in the Format command line. The /1 switch is used for formatting a single-sided disk, and the /8 switch is used to format disks at eight sectors per track.

You may also specify a /S switch to copy the MS-DOS system to the diskette being formatted. These switches (and others) are described under the Format command in Chapter 5.

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

```
Formatting...Format complete

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

Volume labels are useful to identify disks -- they are like a name tag for each disk. When you assign a unique volume label to a disk, you can always be sure that you know which disk you are using. Volume labels are displayed with many DOS commands, including the DIR (directory) command described in this chapter.

Type a volume label in response to the prompt above if you want to identify this disk, and press **RETURN**. An example of a volume label is PROGRAMS.

If you do not want to attach a label to this disk, simply press **RETURN**. You should see on your screen a message similar to this:

```
362,496 bytes total disk space
xxxxxxx bytes used by system
362,496 bytes available on disk

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

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

How to Back Up Your Disks

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

The DISKCOPY Command

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

The format of the DISKCOPY command is:

```
DISKCOPY [drive1:] [drive2:]
```

Drive 1 is the disk drive that contains the disk that you want to copy. Drive 2 is the disk drive that contains the blank or "destination" disk. The blank disk must be formatted prior to running DISKCOPY.

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

```
DISKCOPY A: B:
```

MS-DOS responds:

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

Make sure the MS-DOS disk is in Drive A and insert a blank, formatted disk in Drive B. Next, press any character key and MS-DOS will begin copying the MS-DOS disk. After MS-DOS has copied the disk, MS-DOS displays:

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

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

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

Note: If either of the disks that you are using has defective tracks, DISKCOPY will not work. Use the COPY command to back up your disks in these cases. Refer to Chapter 5 for information on how to use COPY.

How MS-DOS Keeps Track Of Your Files

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

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

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

The DIR (Show Directory) Command

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

```
DIR A:
```

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

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

COMMAND   COM   16276   10-29-81   11:48a
CHKDSK    COM    6272   10-26-82   12:12p
SYS       COM    1400   10-29-82    6:30p
EDLIN     COM    4419    1-01-80   12:41a
RECOVER   COM    2281   10-29-82    5:37p
PRINT     COM    3899   10-27-82   12:19p
FORMAT    COM    5605   10-28-82    9:55a
SORT      EXE    1280   10-27-82    3:18p
MORE      COM     291   10-27-82    3:20p
FIND      EXE    5888   01-01-80   12:57a
CONFIG    SYS      33   10-18-82    5:02p
FC        EXE   10624   10-27-82    7:00p

      12 File(s)       83967 bytes free
```

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

You can also get information about any file on your disk by typing DIR and a filename. For example, if you have created a file named MYFILE.TXT, the following command displays the file's directory information (name of file, size of file, date last edited):

```
DIR MYFILE.TXT
```

More information about the DIR command, including additional options, appears in Chapter 5.

Automatic Program Execution

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

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

Files

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

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

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

The CHKDSK (Check Disk) Command

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

To check the disk in drive A:, type:

```
CHKDSK A:
```

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

DOS Guide

More About Files

More About Files

Overview

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

How to Name Your Files

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

```
NEWFILE.EXE
```

The name of a file consists of two parts: the filename and the extension. In this example, 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 up to three characters. You can type any filename in upper or lowercase letters and MS-DOS translates the name into uppercase characters.

In addition to the filename and the filename extension, the file specification may include a drive designation. A drive designation tells MS-DOS to look on the disk in the designated drive to find the filename typed.

For example, to find directory information about the file NEWFILE.EXE which is located on the disk in Drive A (and Drive A is NOT the default drive), type the following command:

```
DIR A:NEWFILE.EXE
```

Directory information about the file NEWFILE.EXE is now displayed on your screen. If Drive A is the default drive, MS-DOS searches only the disk in Drive A for the filename NEWFILE and so the drive designation is not necessary. A drive designation is needed if you want to tell MS-DOS to look on the other drive to find a file.

Your filenames will probably be made up of letters and numbers, but other characters are allowed, too. Legal characters for filename extensions are the same as those for filenames.

The complete list of the characters you can use in filenames and extensions is as follows:

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

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

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

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

Examples of file specifications are:

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

Wild Cards

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

The ? Wild Card

A question mark (?) in a filename or filename extension indicates that any character can occupy that position. For example, the following MS-DOS command lists all directory entries on the default drive that have 8 characters, begin with TEST, have any next character, end with the letters RUN, and have a filename extension of .EXE:

```
DIR TEST?RUN.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
```

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

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

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

Examples:

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

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

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

```
DIR B:*.TXT
```

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

Illegal Filenames

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

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

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

LST or
PRN Used when referring to the printer device.

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

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

How to Copy Your Files

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

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

The format of the COPY command is:

COPY filespec [filespec]

For example, the following command copies the file MYFILE.TXT on the disk in Drive A to a file named MYFILE.TXT on the disk in Drive B:

COPY A:MYFILE.TXT B:MYFILE.TXT

Duplicate copies of MYFILE.TXT now exist on separate diskettes.

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

```
COPY A:MYFILE.TXT A:NEWNAME.TXT
```

You now have two copies of your file on disk A, one named MYFILE.TXT and the other named NEWNAME.TXT.

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

How to Protect Your Files

MS-DOS is a powerful and useful tool in processing your personal and business information. As with any information system, inadvertent errors may occur and information may be misused.

If you are processing information that cannot be replaced or requires a high level of security, you should take steps to ensure that your data and programs are protected from accidental or unauthorized use, modification, or destruction. Simple measures you can take (such as removing your disks when they are not in use, keeping backup copies of valuable information, and installing your equipment in a secure facility) can help you maintain the integrity of the information in your files.

Directories

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

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

In an office, you can separate files by putting them in different filing cabinets. This procedure, in effect, creates different directories of information. MS-DOS allows you to organize the files on your disks into directories. Directories are a way of dividing your files into convenient groups of files. For example, you may want all of your accounting programs in one directory and text files in another.

Any one directory can contain related files, and may also contain other directories (referred to as subdirectories). This method of organizing your files is called a hierarchical directory structure.

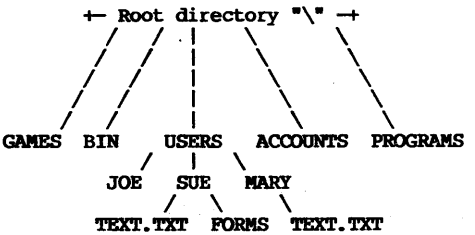
A hierarchical directory structure can be thought of as a "tree" structure, where directories are the branches of the tree and files are the leaves. This type of "tree" grows downward; that is, the "root" is at the top.

The root is the first level in the directory structure. When you format a disk, the root directory is automatically created and named the backslash character (\). You can create additional subdirectories by following the instructions in Chapter 4.

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

It is possible to find any file in the system by starting at the root and traveling down any of the branches to the desired file. Conversely, you can start where you are within the file system and travel towards the root.

Each directory stores its own filenames, independent of all other directories. Unless you take special action when you create a file, the new file is created in the directory in which you are currently working. Files of the same name can exist independently on a disk in different directories.



The root directory (\) is the first level in the directory structure. You can create subdirectories from the root by using the MKDIR command (refer to Chapter 5 for information on MKDIR).

In this example, five subdirectories have been created. These include:

1. A directory of games, named GAMES
2. A directory of all external commands, named BIN
3. A USER directory containing separate subdirectories for all users of the system
4. A directory containing accounting information, named ACCOUNTS
5. A directory of programs, named PROGRAMS

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

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

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

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

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

```
DIR \USER\MARY
```

To list the files and subdirectories in the root directory, type:

```
DIR \
```

Filenames and Paths

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

Pathnames

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

The syntax of pathnames is:

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

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

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

****	Indicates the root directory.
\PROGRAMS	Sample directory under the root directory containing program files.
\USER\MARY\FORMS\1A	A typical full pathname. This one happens to be a file named 1A in the directory named FORMS belonging to the USER named MARY.
USER\SUE	A relative pathname; it names the file or directory SUE in the subdirectory USER of the working directory. If the working directory is the root (\), the complete "actual" pathname is \USER\SUE.
TEXT.TXT	Name of a file or directory in the working directory.

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

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

.. The shorthand name of the working directory's parent directory. If you type:

```
DIR ..
```

MS-DOS lists the files in the parent directory of your working directory. If you type:

```
DIR ..\..
```

MS-DOS lists the files in the parent's PARENT directory.

Pathing and External Commands

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

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

For example, if you are in a working directory named \USER\SUE and all MS-DOS external commands are in the root directory, you must tell MS-DOS to search the root to find external commands. The following command instructs MS-DOS to search in your working directory and the root directory for all commands:

```
PATH \
```

To specify searching for commands on Drive A's root directory, type **PATH A:**. If all external commands are located in a subdirectory, enter **PATH \<subdir>** where "subdir" is the name of the subdirectory which contains the commands.

You only have to specify the path command once to MS-DOS during your terminal session. If you want to know what the current path is, type **PATH** and the current value of PATH is printed.

You can cause the PATH command to execute automatically at each system boot by placing the PATH command into an AUTOEXEC.BAT file as described in Chapter 4.

See Chapter 5 for more information on the MS-DOS PATH command.

Pathing and Internal Commands

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

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

The syntax of these four commands is shown below.

COPY <pathname pathname>	If the second pathname to COPY is a directory, all files are copied into that directory.
DEL <pathname>	If the pathname is a directory, all the files in that directory are deleted. **Note:** The prompt "Are you sure (Y/N)?" is displayed if you try to delete a path. Type **Y** to complete the command, or type **N** for the command to abort.
DIR <pathname>	Displays the directory for a specific path.
TYPE <pathname>	You must specify a file in a path for this command. MS-DOS displays the file on your screen in response to the TYPE pathname command.

Displaying Your Working Directory

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

```
CHDIR
```

displays:

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

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

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

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

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

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

Creating a Directory

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

```
MKDIR NEWDIR
```

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

To create files for the new directory, use the MS-DOS line editor, EDLIN. Chapter 7 describes how to use EDLIN to create and save files.

How to Change Your Working Directory

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

```
A>CHDIR \USER
```

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

How to Remove a Directory

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

```
RMDIR NEWDIR
```

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

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

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

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

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

DOS Guide

Learning More About Commands

Learning About Commands

Overview

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

- Compare, copy, display, delete, and rename files
- Copy and format disks
- Execute system programs such as EDLIN, as well as your own programs
- Analyze and list directories
- Enter date, time, and remarks
- Set various printer and screen options
- Copy MS-DOS system files to another disk
- Request MS-DOS to wait for a specific period of time

Types of MS-DOS Commands

There are two types of MS-DOS commands: Internal and External.

Internal commands are the simplest, most commonly used commands. You cannot see these commands when you do a directory listing on your MS-DOS disk because they are part of the command processor. When you type these commands, they execute immediately.

The following internal commands are described in Chapter 5. Synonyms are shown in parentheses:

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

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

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

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

ARCHIVE (ARK)	EXE2BIN	FORMAT	PRINT
CHKDSK	EXPAND	HDFORMAT	RECOVER
CONVERT	FC	LINK	SIZE
DEBUG	FILETYPE	MAKEDB	SORT
DISKCOPY	FIND	MORE	SYS
EDLIN	FIXDISK	MOVAFILE	TREE

Command Options

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

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

Command [options...]

where options may be:

d: Refers to disk drive designation.

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

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

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

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

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

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

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

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

Information Common to All MS-DOS Commands

The following information applies to all MS-DOS commands:

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

   DEL MYFILE.OLD NEWFILE.TXT
   RENAME,AFILE BFILE

   You can also use the semicolon (;), the equal sign (=), or the tab key as delimiters in MS-DOS commands. In this manual, a space is used as the delimiter in commands.
4. Do not separate a file specification with delimiters, since the colon and the period already serve as delimiters.
5. When instructions say "Press any key," you can press any alpha (A-Z) or numeric (0-9) key.
6. You must include the filename extension when referring to a file that already has a filename extension.
7. You can abort commands when they are running by pressing **CTRL C**.
8. Commands take effect only after you have pressed the **RETURN** key. When the screen prompts "press **ENTER**", press **RETURN**.
9. Wild cards (global filename characters) and device names (for example, PRN or CON) are not allowed in the names of any commands.

10. When commands produce a large amount of output on the screen, the display automatically scrolls to the next screen. You can press **CTRL S** to suspend the display. Press any key to resume the display on the screen.

11. MS-DOS editing and function keys can be used when entering commands. Refer to Chapter 6 for a complete description of these keys.

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

13. Disk drives are referred to as source drives and destination drives. A source drive is the drive you are transferring information from. A destination drive is the drive you are transferring information to.

Batch Processing

Often you may find yourself typing the same sequence of commands over and over to perform some commonly used task. With MS-DOS, you can put the command sequence into a special file called a batch file, and execute the entire sequence simply by typing the name of the batch file.

"Batches" of your commands in such files are processed as if they were typed at a terminal. Each batch file must be named with the extension .BAT, and is executed by typing the filename without its extension.

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

Batch processing is useful if you want to execute several MS-DOS commands with one batch command. The following batch file, created using EDLIN, formats and checks a new disk:

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

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

```
NEWDISK
```

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

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

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

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

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

4. If you press **CTRL C** while in batch mode, this prompt appears:

   Terminate batch job (Y/N)?

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

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

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

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

The AUTOEXEC.BAT File

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

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

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

If MS-DOS does not find an AUTOEXEC.BAT file when you first load the MS-DOS disk, the date and time prompts are issued.

How to Create an AUTOEXEC.BAT File

An AUTOEXEC.BAT file may be created by using either EDLIN or the COPY command. The following procedure uses COPY to create the file.

If you wanted to automatically designate a path each time you started MS-DOS, you could create an AUTOEXEC.BAT file as follows:

1. Type:

   COPY CON AUTOEXEC.BAT

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

   Note: If you already have an AUTOEXEC.BAT file, the COPY CON procedure will overwrite it and your file will be replaced.

2. Now type:

   ```
   B:
   PATH A:\
   ```

   These statements instruct MS-DOS to make Drive B the logged drive and to search for external commands in the root directory of Drive A.

3. Press **CTRL Z** and the **RETURN** key to put these commands in the AUTOEXEC.BAT file. The commands will now run automatically whenever you start MS-DOS.

You can enter any MS-DOS command or series of commands in the AUTOEXEC.BAT file.

Note: If you use an AUTOEXEC.BAT file, MS-DOS does not prompt you for a current date and time unless you include the DATE and TIME commands in the AUTOEXEC.BAT file.

Creating a .BAT File with Replaceable Parameters

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

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

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

```
A>COPY CON RENTEXT.BAT
RENAME %1 %2.TXT
ERASE %1.BAK
TYPE %2.TXT
TYPE %0.BAT
```

Now, press **CTRL Z** and **RETURN**. MS-DOS responds with this message:

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

The file RENTEXT.BAT, which consists of four commands, now resides on the disk in the default drive. **Note:** If you already have a file named RENTEXT.BAT on the default disk, this procedure will replace it.

The dummy parameters %1 and %2 are replaced sequentially by the parameters you supply when you execute the file. The dummy parameter %0 is reserved to represent the batch file's specifications (drive designator and filename), which in this example is RENTEXT).

Notes:

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

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

3. The value which a dummy parameter represents may be any number of characters in length.

Executing a .BAT File

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

The dummy parameter %0 is replaced by the batch file specifications. Values for all other parameters must be entered sequentially on the command line.

For example, remember that the file RENTEXT.BAT consists of four lines:

```
RENAME %1 %2.TXT
ERASE %1.BAK
TYPE %2.TXT
TYPE %0.BAT
```

To execute the RENTEXT batch process, type the values you wish to supply for the sequential variables:

```
RENTEXT BOILER 1-INTRO
```

RENTEXT is substituted for %0, BOILER for %1, and 1-INTRO for %2.

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

```
RENAME BOILER 1-INTRO
ERASE BOILER.BAK
TYPE 1-INTRO.TXT
TYPE RENTEXT.BAT
```

Input and Output

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

In addition, "pipes" can be created that allow output from one command to become the input to another. Redirection and pipes are discussed in the next sections.

Redirecting Your Output

Most commands produce output that is sent to your terminal. You can send this information to a file by using a greater-than sign (>) in your command. For example, the following command displays a directory listing of the disk in the default drive on the terminal screen:

```
DIR
```

The same command can send this output to a file named MYFILES by designating the output file on the command line:

```
DIR >MYFILES
```

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

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

```
DIR >>MYFILES
```

If MYFILES does not exist, it is created.

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

For example, the following command sorts the file MYFILES and sends the sorted output to a file named LIST1:

```
SORT <MYFILES >LIST1
```

Filters

A filter is a command that reads your input, transforms it in some way, and then outputs it, usually to the display screen or 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 several commands. MS-DOS filters include FIND, MORE, and SORT. Their functions are described in detail in Chapter 5 and are listed below:

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

Command Piping

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

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

```
DIR | SORT
```

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

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

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

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

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

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

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

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

```
DIR | SORT | MORE
```

This command sorts your directory and displays it one screen at a time, putting --MORE-- at the bottom of your screen when there is more output to be seen.

DOS Guide

DOS Commands

MS-DOS Commands

Overview

The following MS-DOS commands are described in this chapter. Synonyms for commands are enclosed in parentheses.

ARCHIVE (**ARK**) Backs up and restores files on the hard disk

BREAK Checks for a CTRL C entered at the keyboard

CHDIR (**CD**) Changes directories or displays working directory

CHKDSK Scans the directory of the default or designated drive and checks for consistency

CLS Clears the screen

CONVERT Bi-directional CP/M to MS-DOS file converter

COPY Copies file(s) as specified

CTTY Changes console TTY

DATE Displays and sets date

DEBUG Program debugging tool

DEL (**ERASE**) Deletes file(s) as specified

DIR Lists requested directory entries

DISKCOPY Copies entire diskettes

ECHO Turns batch file echo feature on or off

EDLIN Line editor for creating source or text files

EXE2BIN Converts executable files to binary format

EXIT Exits COMMAND.COM and returns to previous command processor

EXPAND Creates several lines from one list of arguments

FC Compares files

FILETYPE Changes or displays file and directory attributes

FIND Searches files for a specified text string

FIXDISK Locates bad sectors on the hard disk

FOR Batch command extension for repeated execution of DOS commands

FORMAT Formats a diskette for the MS-DOS operating system

GOTO Batch command extension to transfer control to a specified line in a batch file

HDFORMAT Formats the hard disk

IF Batch command extension to allow conditional execution of commands

LINK Combines separately produced object modules

MAKEDB Converts hex file values to decimal values

MKDIR (MD) Makes a directory

MORE Displays output one screen at a time

MOVAFILE Transfers files with the archive attribute set

PATH	Specifies the directory which contains external commands
PAUSE	Suspends execution in a batch file
PRINT	Queues and prints text files
PROMPT	Designates command prompt
RECOVER	Recovers files from a damaged disk
REM	Displays a comment in a batch file
REN (RENAME)	Renames first file as the second file
RMDIR (RD)	Removes a directory
SET	Sets one string value equivalent to another
SHIFT	Allows access to more than 10 batch replaceable parameters
SIZE	Displays size of specified files and total size of all files
SORT	Sorts data alphabetically, forward or backward
SYS	Transfers MS-DOS system files to the specified drive
TIME	Displays and sets time
TREE	Displays directory paths
TYPE	Displays the contents of the specified file
VER	Prints MS-DOS version number

VERIFY Verifies that a file has been written to disk correctly

VOL Prints volume identification label

Syntax Notation

The following notation describes the correct format for entering MS-DOS commands:

1. Any words shown in capital letters are keywords and must be entered exactly as shown. You can enter keywords in any combination of upper/lowercase letters.

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

3. Items in square brackets ([ ]) are optional. Include optional information if desired (do not include the square brackets).

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

5. A bar (|) indicates an OR statement in a command where you must select one of the items on either side of the bar.

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

ARCHIVE or ARK - Back Up Files

TYPE External

PURPOSE Archives (backs up) files from a fixed disk to diskettes, restores files from diskettes to a fixed disk, or lists files on a disk.

SYNTAX ARCHIVE [s:] [path] [filename] [.ext] d:/B|/R|/F [/S] [/M] [/A] [/P] [/D:mm-dd-yy]

REMARKS Use only MS-DOS formatted diskettes when using ARCHIVE.

The first parameter you specify is the source disk and optional filespec to archive, restore, or list. If the filespec is not included, only the files in the current directory are used.

The second parameter is the destination disk you wish to archive to or restore from. The second parameter must be a drive letter followed by a colon. Immediately following the colon is the current switch which defines the operation to be performed (backup, restore, or file list). Finally, one or more optional parameters may be entered.

Global filename characters are allowed in the filename or extension. For example:

```
ARCHIVE C:*.ASM A:/B
```

All files in the current directory with the extension of .ASM are archived onto the Drive A diskette.

Command Switches

/B The /B switch copies files from the source disk to the destination diskette. When a disk is filled you are prompted to insert another diskette. This switch also splits large files into smaller files as needed to allow them to be saved on diskettes. The Archive attribute is removed from files by the /B switch. New or modified files automatically have the Archive attribute set.

/R The /R switch restores archived files from diskettes back to their original file directories on the hard disk. Restore automatically recreates any missing subdirectories.

/F The /F switch allows you to display a directory of files contained on the source disk.

Optional Parameters

/S The /S parameter causes the files in all subdirectories to be acted upon by all commands. Only the subdirectories at all levels beyond the specified directory are searched.

/M The /M parameter indicates that only files that have been modified since the last archive are used.

/A The /A parameter indicates that archived file(s) should be added to the files on an already existing ARK diskette. If this parameter is omitted, all files on the destination diskette are deleted prior to archiving the new files.

/D:mm-dd-yy The /D parameter can be used to archive or list files that have been modified since a specific date.

/P The /P parameter displays prompts during restore operations for read-only and modified disk files, requesting confirmation for copying over those files.

Examples

The following command is used to archive the entire disk including all subdirectories on Drive C to the diskettes on Drive A:

```
ARCHIVE C: A:/B /S
```

To restore all of the archived files in the example above, use the following command:

```
ARCHIVE A: C:/R /S
```

From this point on you could do an incremental save of disk C with this command:

```
ARCHIVE C: A:/B /S /M
```

BREAK

TYPE Internal

PURPOSE Instructs MS-DOS to check for a CTRL C being entered at the keyboard.

SYNTAX BREAK ON|OFF

REMARKS If you run an application program that uses **CTRL C** function keys, specify **BREAK OFF** to turn off the MS-DOS CTRL C function so that when you press **CTRL C** you affect your program and not the operating system.

Specify **BREAK ON** when you have finished running your application program and return to MS-DOS. Typing **BREAK** without parameters displays the current BREAK status on the screen.

CHDIR or CD - Change Directory

TYPE Internal

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

SYNTAX CHDIR [pathname][..]

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

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

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

```
CHDIR ..
```

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

CHDIR used without a pathname displays your working directory. If your working directory is \USER\JOE on Drive B and you type **CHDIR** and press **RETURN**, MS-DOS displays:

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

To get to the root directory, type:

```
CHDIR \
```

If your working directory is the root directory, entering **CHDIR** displays:

```
B:\
```

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

CHKDSK - Check Disk

TYPE External

PURPOSE Scans the directory and File Allocation Table of the specified disk drive and produces a disk and memory status report.

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

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

A sample status report follows:

```
362496  bytes total disk space
 52224  bytes in 3 hidden files
  1024  bytes in 2 directories
305152  bytes in 28 user files
  3072  bytes available on disk

262144  bytes total memory
230448  bytes free
```

CHKDSK does not correct the errors found in your directory unless you specify the /F (fix) switch. Typing /V (verify) causes CHKDSK to display a series of messages indicating its progress and provide more detailed information about the errors it finds.

You can redirect the output from CHKDSK to a file by typing:

```
CHKDSK A:>filename
```

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

Error Handling

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

Invalid drive specification

Invalid parameter

Invalid subdirectory entry

Cannot CHDIR to <filename>

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

<filename> contains
non-contiguous blocks

All specified file(s) are contiguous

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

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

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

2. **Invalid current directory**
Processing cannot continue

Recovery: Restart the system and rerun CHKDSK.

3. **Cannot CHDIR to root**
Processing cannot continue

Recovery: The disk you are checking is bad. Try restarting MS-DOS and RECOVER the disk.

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

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

5. **X lost clusters found in y chains**
Convert lost chains to files (Y/N)?

Recovery: If you respond **Y** to this prompt, CHKDSK creates a directory entry and a file for each cluster (files created by CHKDSK are named FILEnnnn). CHKDSK then displays:

X bytes disk space freed

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

X bytes disk space would be freed

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

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

7. **Insufficient room in root directory**
Erase files in root and repeat CHKDSK

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

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

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

CLS - Clear Screen

TYPE Internal

PURPOSE Clears the display screen.

SYNTAX CLS

REMARKS The CLS command sends the ANSI escape sequence ESC[2J to your console, which clears your display screen.

CONVERT

TYPE External

PURPOSE Copies CP/M files to MS-DOS format or MS-DOS files to CP/M format.

SYNTAX CONVERT

REMARKS CONVERT allows you to access, convert, and copy files on a CP/M diskette while under the MS-DOS operating system. You can also display a CP/M directory or text file using the CONVERT utility.

CP/M files must be created on and copied to diskettes that have been formatted for CP/M. If you are converting an MS-DOS file to CP/M, the destination diskette must already be formatted for CP/M before you use CONVERT.

To use the CONVERT utility, type **CONVERT.** A menu with the following options is then displayed:

List CP/M Directory -- Lists the directory of a CP/M format diskette.

DOS to CP/M -- Converts a file from the MS-DOS format to a specified CP/M format, renames the file extension from .CPM to .COM, and copies the file to a CP/M format diskette.

CP/M to DOS -- Converts a file from a specified CP/M format to the MS-DOS format, renames the file extension from .COM to .CPM, and copies the file to the MS-DOS disk or diskette.

Type CP/M File -- Displays the contents of a CP/M text file. One screenful of text is displayed at a time.

Use the right and left arrow keys to move to the desired option and press **RETURN.** To exit CONVERT, press the **ESC** key or move to the EXIT option and press **RETURN.**

If you have selected an option other than EXIT, another menu is displayed. This menu requires you to specify the CP/M disk and the file or files on which the selected task is to be performed as described below.

For the prompt **CP/M Drive:**, supply the letter **A** or **B** to indicate which drive contains the CP/M format diskette.

For **File Name:**, supply a 1-to-8 character filename. The filename optionally may be preceded with a path or followed by a 1-to-3 character extension. If you press **RETURN** without typing a filename, all files are selected (default is *.*). Wildcards may be used in a filename.

Note: Drive specifications for filenames are ignored in the CONVERT utility. CONVERT correctly selects the drive which contains the CP/M or MS-DOS file automatically for each option.

For the final prompt, you must select the CP/M diskette type (the type of system which formatted your CP/M diskette). Use the down arrow key to move to the appropriate CP/M diskette type and press **RETURN.**

Examples:

In each of the following examples, assume you have an MS-DOS format disk in Drive A and a CP/M format disk in Drive B. The CONVERT utility is on the MS-DOS disk in Drive A.

1. Select **List CP/M Directory** on the main menu and press **RETURN.** To get a directory listing of only the TXT files on your CP/M diskette, respond to the prompts as follows:

   CP/M Drive: B
   Filename: *.TXT
   Also Select a CP/M Diskette Type (select **Otrona**)

2. Select the **DOS to CP/M** option on the main menu. If you wish to convert an MS-DOS file called USER.COM in the path USERLIB to CP/M, respond as shown below:

   CP/M Drive: B
   Filename: \USERLIB\USER.COM
   Also Select a CP/M Diskette Type (select **Otrona**)

3. Select **CP/M to DOS** on the main menu. If you wish to convert all CP/M COM files to MS-DOS, you could respond:

   CP/M Drive: B
   Filename: *.COM
   Also Select a CP/M Diskette Type (select **Otrona**)

4. Select the **Type CP/M File** option on the main menu. If you wish to type a text file called JEFF, respond to the prompts as shown below:

   CP/M Drive: B
   Filename: JEFF
   Also Select a CP/M Diskette Type (select **Otrona**)

   Note in each case that the drive specification (A or B) is unnecessary.

COPY

TYPE Internal

PURPOSE Copies one or more files to the same disk or another disk. Optionally, you can give the copies different names.

SYNTAX COPY [CON] <filespec> [filespec] [/V]

REMARKS The CON option is used to create a file directly from console input and is described later in this section. The first <filespec> is the source file, and the second <filespec> is the destination file.

The filespec may contain a pathname. If the filespec specifies a pathname only, all files within that path (subdirectory) are copied to the file or path (subdirectory) specified as the second filespec.

If the second filespec option is not given, the copy is made on the default drive and has the same name as the original file. If the first filespec is on the default drive and the second filespec is not specified, the COPY is aborted (copying files to themselves is not allowed). MS-DOS displays the error message:

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

The second option [filespec] may take three forms:

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

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

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

The /V switch verifies that the sectors written on the destination disk are recorded properly. Although recording errors are rare, you can verify that critical data has been correctly recorded if you wish. This option causes the COPY command to run slower because MS-DOS checks each entry recorded on the disk.

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

For example:

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

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

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

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

This command takes all files with a filename extension of .LST and combines them into a file named COMBIN.PRN.

In the following example, each file that matches *.LST is combined with the corresponding .REF file to form a file with the same filename but with the extension .PRN. Thus, FILE1.LST is combined with FILE1.REF to form FILE1.PRN, XYZ.LST is combined with XYZ.REF to form XYZ.PRN, and so on.

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

The following COPY command combines all files which match *.LST and *.REF into one file named COMBIN.PRN:

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

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

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

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

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

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

The error is not detected, however, until ALL.LST is appended. At this point it may already be destroyed.

You may copy input directly from a specified device into a file using the format COPY CON <filespec>. For example, if the console is your display device, you can create a file by typing:

```
COPY CON FILE1.BAT
```

You then type each line as you wish it to appear in a file (upper/lowercase is preserved). End your file by typing **CTRL Z** and **RETURN**. The lines which appear on your screen are then saved into the file **FILE1.BAT**.

Warning: If you specify COPY CON for an already existing file, the new text you type will replace the text already saved in the file. No warning is issued.

CTTY - Change Console Device

TYPE Internal

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

SYNTAX CTTY <device>

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

For example, the command:

```
CTTY AUX
```

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

```
CTTY CON
```

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

DATE

TYPE Internal

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

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

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

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

Press **RETURN** if you do not want to change the date shown. DOS automatically supplies the day of the week whenever you specify a date.

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

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

In this case, you do not have to answer the "Enter new date" prompt. The new date must be numeric (letters are not permitted). Valid options are:

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

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

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

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

DATE then waits for you to enter a valid date.

DEBUG

TYPE External

PURPOSE Debugging program for binary and executable object files.

SYNTAX DEBUG [<filespec>]

REMARKS DEBUG is a debugging program that provides a controlled testing environment for binary and executable object files. DEBUG eliminates the need to reassemble a program to see if a problem has been fixed by a minor change. DEBUG allows you to alter the contents of a file or CPU register and then immediately reexecute a program to check the validity of the changes.

DEBUG may be started by typing **DEBUG** and waiting for the hyphen (-) prompt, or you may supply the name of the file to be debugged. DEBUG commands include Assemble, Compare, Dump, Enter, Fill, Go, Hex, Input, Load, Move, Name, Output, Quit, Register, Search, Trace, Unassemble, and Write.

All DEBUG commands may be aborted at any time by pressing **CTRL C**. Pressing **CTRL S** suspends the display so that you can read the screen 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 MS-DOS command level.

For details on using DEBUG and DEBUG commands, see the Programmer's Reference Guide.

DEL or ERASE - Delete File

TYPE Internal

PURPOSE Deletes all files with the designated filespec.

SYNTAX DEL [pathname][filespec]

REMARKS If the filespec is *.*, the prompt "Are you sure (Y/N)?" appears. If a **Y** or y is typed as a response, all files are deleted as requested. The wild card characters question mark (?) and asterisk (*) are permitted in the DEL command. You can also type **ERASE** for this command.

If you specify a pathname only for the filespec, the prompt "Are you sure (Y/N)?" appears. A **Y** reply will erase all files within the path, but will not erase the path itself.

DIR - Show Directory

TYPE Internal

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

PURPOSE Lists the files in a directory.

REMARKS If you type **DIR** without options, all entries under the current directory on the default drive are listed.

If only the drive specification is given (DIR d:), all entries on the disk in the current directory of the specified drive are listed. If only a filename is entered with no extension (DIR filename), all files with the designated filename in the current directory of the disk in the default drive are listed. If you designate a file specification (for example, DIR d:filename.ext), all files with the specified filename in the current directory on the specified drive are listed.

If you specify a pathname instead of a file, all files within that path (subdirectory) are listed. A pathname specified with a filename lists only the matching file(s). 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 question mark (?) and asterisk (*) may be used in the filename option. For your convenience, the following DIR commands are equivalent:

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

Two switches, /P and /W, may be specified with DIR. The /P switch selects Page Mode, which causes the display of the directory to pause after the screen is filled. To resume display of output, press any key.

The /W switch selects Wide Display. /W displays five filenames per line, without other file information. For example:

```
COMMAND COM   CHKDSK  COM   FORMAT  COM   TELINK  EXE   MORE    COM
FIXDISK COM   EDLIN   COM   PRINT   TXT   RECOVER COM   DISPLAY TXT
```

DISKCOPY

TYPE External

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

SYNTAX DISKCOPY [s:][d:]

REMARKS The first option you specify is the source drive [s:]. The second option is the destination drive [d:]. The disk in the destination drive must be formatted prior to using DISKCOPY.

You can specify the same drives or different drives. If the same drive is designated, a single-drive copy operation is performed. In each case, you are prompted to insert the disk(s) at the appropriate time. DISKCOPY waits for you to press any key before continuing.

After copying a disk, DISKCOPY prompts:

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

If you press **Y**, another copy is performed on the same drive(s) that you originally specified, after you are prompted to insert the proper disks.

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

Notes:

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

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

3. Both disks must have the same number of physical sectors and those sectors must be the same size.

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

   A fragmented disk can cause poor performance due to delays involved in finding, reading, or writing a file. Use the COPY command instead of DISKCOPY to copy a fragmented disk to eliminate the fragmentation.

For example:

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

copies all files from the disk in Drive A to the disk in Drive B.

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

6. If disk errors are encountered during a DISKCOPY, MS-DOS displays:

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

Refer to Appendix C for information on this error message.

ECHO

TYPE Internal

PURPOSE Turns batch echo feature on and off.

SYNTAX ECHO [ON|OFF|message]

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

Typing a message after **ECHO** echoes the message on your screen, even if you have specified **ECHO OFF**. Quote characters around the message are unnecessary.

If **ECHO** is typed without parameters, the current ECHO setting is displayed.

ECHO is ON after a powering up or resetting the system. If you have a batch file that turns ECHO OFF, such as:

```
ECHO OFF
DIR A:
```

the prompt (such as A>) is turned off when each command is executed in the batch file. ECHO is set back to ON automatically after the batch file completes execution.

EDLIN

TYPE External

PURPOSE Line editor to create, change, and display source program files or text files.

SYNTAX EDLIN <filespec>

REMARKS EDLIN, the line editor program, can be used 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; or search for, delete, or replace text in one or more lines.

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

When you insert lines, all line numbers following the inserted text increase 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.

To start EDLIN, type:

```
EDLIN <filespec>
```

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

```
New file
*_
```

The prompt for EDLIN is an asterisk (*).

Special control characters, switches, and commands for EDLIN are described in Chapter 7.

EXE2BIN - Convert .EXE Files

TYPE External

PURPOSE Converts .EXE files to binary format to save disk space and load programs faster.

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

REMARKS This command converts .EXE files to binary format. The file named by <filespec> is the input file. If no extension is specified, it defaults to .EXE. The input file is converted to .COM file format (memory image of the program) and placed in the output file.

If you do not specify a drive, the drive of the input file is used. If you do not specify an output filename, the input filename is used. If you do not specify a filename extension in the output filename, the new file is given an extension of .BIN.

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

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

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

2. If CS:IP is specified as 0000:100H, it is assumed that the file is to be run as a .COM file with the location pointer set at 100H by the assembler statement ORG and the first 100H bytes of the file are deleted. No segment fixups are allowed because .COM files must be segment relocatable.

   Once the conversion is complete, you may rename the resulting file with a .COM extension. The command processor then is able to load and execute the program the same as the .COM programs supplied on your MS-DOS disk.

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

File cannot be converted

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

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

File not found -- The file is not on the specified disk.

Insufficient memory -- There is not enough memory to run EXE2BIN.

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

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

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

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

WARNING - Read error on EXE file. -- The amount read is less than the size in the header. This is a warning message only.

EXIT

TYPE Internal

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

SYNTAX EXIT

REMARKS This command can be used when you are using a user-written command processor in addition to COMMAND.COM. EXIT allows you to exit your user-written command processor, use COMMAND.COM, and return to your user-written command processor.

For example, to look at a directory on Drive B while using another command processor, start COMMAND.COM by typing **COMMAND** in response to the default drive prompt:

```
A>COMMAND
```

You can now type the **DIR B:** command and MS-DOS displays the directory. When you type **EXIT**, you return to your user-written command processor (the previous level).

EXPAND

TYPE External

PURPOSE Allows you to create several command lines from one set of arguments.

SYNTAX EXPAND <outfile> <firstarg> <filespec> [<secondarg>]

REMARKS EXPAND is a time-saving command which allows you to enter one EXPAND command to generate command lines for an entire group of files. It creates in a specified <outfile> a number of lines that contain a copy of <firstarg>, followed by one match of <filespec>, followed by <secondarg>, if it exists.

<Outfile> can be any new filename with the extension .BAT. If an existing filename is used for <outfile>, the existing file is replaced with <outfile>.

For example, if you type:

```
EXPAND ASMALL.BAT  MASM *.ASM ,,PRN;
```

the file ASMALL.BAT now contains the following commands:

```
MASM A.ASM ,,PRN;
MASM B.ASM ,,PRN;
MASM C.ASM ,,PRN;
```

The entire group of files can be assembled by typing **ASMALL.**

FC

TYPE External

PURPOSE Compare two files.

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

REMARKS The File Comparison (FC) utility compares the contents of two files. For example, you may want to compare two copies of a file to see which one is current. The differences between the two files can be printed to the console or to a third file. The files may be source files or binary files.

Comparisons may be made on a line-by-line or a byte-by-byte basis. The line-by-line comparison isolates and prints blocks of lines that differ between two files. The byte-by-byte comparison displays bytes that differ between files.

When comparing two files, FC matches the first file (filename1) against the second (filename2) and reports any differences between them. Both filenames can be pathnames, as shown in the following example:

```
FC B:\TEXT\MEMO\FILE1.TXT \MEMO\FILE2.TXT
```

FC takes FILE1.TXT in the \TEXT\MEMO directory of disk drive B: and compares it with FILE2.TXT in the \MEMO directory. Because no drive is specified for filename2, FC assumes that the \MEMO directory is on the disk in the default drive.

Switches for FC are briefly described below:

/# Specifies the number of lines required to match before the files are considered to match again after a mismatch, where # is a number from 0 to 9.

/B Forces a binary comparison of both files. Files are compared byte-to-byte, with no attempt to re-synchronize after a mismatch.

/W Compresses tabs and spaces (whites) during the comparison so contiguous whites in a line are considered one space.

/C Ignores the case of letters.

Details on the File Comparison utility, including switches and examples, are described in Chapter 8.

FILETYPE

TYPE External

PURPOSE Change and display file and directory attributes.

SYNTAX FILETYPE [pathname/]<filespec> [+|-A][+|-R][+|-H][+|-S]

REMARKS FILETYPE is used to change and display file and directory attributes. If you specify **FILETYPE** without options or run it improperly, it displays its options and switches and tells you how to operate it. FILETYPE works only with version 2.00 DOS.

FILETYPE displays the following attributes:

A	**Archive**
R	**Read only**
H	**Hidden**
S	**System**
<dir>	**Directory**

You can set or clear any attributes on any file except the <dir> (directory) attribute. If you attempt to change an attribute on a <dir>, FILETYPE displays the attribute change with the message *** Illegal attribute change.** The next FILETYPE display shows the directory without the illegal attribute(s).

Wild cards are acceptable for <filespec>. One or more of the attributes above (A, S, R, or H) may be entered. The pathname must be specified with a forward slash (/) instead of the usual backslash (\) for this command.

Typing **FILETYPE *.*** defaults to displaying attributes for the current directory. To list the attributes of files in another directory, specify a pathname. For example, to list files in the root directory, type **FILETYPE /*.***.

Plus signs (+) set and minus signs (-) remove specified attributes. The default is a plus sign (+) to set attributes. Spaces are ignored.

For example, the following options all set the Archive attribute:

```
A
+A
+ A
```

The following example clears the Archive attribute and sets Read only and Hidden:

```
-A+RH
-A +RH
-A R +H
```

Entering <file> without options lists all of the attributes for the files without changing them. If you attempt an illegal attribute change (such as specifying a non-existent attribute), FILETYPE displays a list of valid parameters and options.

The following command:

```
FILETYPE USER/*.*
```

produces the output listed below:

```
user/                    .  <dir>  . . . . .
user/                   ..  <dir>  . . . . .
user/                 mary  <dir>  . . . . .
user/             file.asm          A . . . .
user/             stat.com          A R . . .
user/           report.txt          . . H . .
user/              listing          A . . . .
 ...                 ...                ...
```

In the example above, MARY is a subdirectory under USER. Because MARY is a directory, its attributes cannot be altered.

FIND

TYPE External

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

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

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

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

Switches for FIND are:

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

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

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

The string should be enclosed in quotes. For example:

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

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

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

displays all names of the files on the disk in drive B: which do not contain the string DAT. Type double quotes around a string that already has quotes in it. Note that FIND searches for an exact match, including upper or lowercase letters. When searching for a filename, you must enter the string in uppercase.

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

Incorrect DOS version -- FIND runs on versions of MS-DOS that are 2.0 or higher.

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

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

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

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

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

FIXDISK

TYPE External

PURPOSE FIXDISK reads all sectors of the hard disk and displays the number of bad blocks (a block consists of two consecutive sectors).

SYNTAX FIXDISK C:

REMARKS FIXDISK automatically maps out (i.e. marks as unusable) any bad sector in an unused portion of the hard disk. However, FIXDISK does not automatically map out any sector that contains data. To map out a bad used sector, you must choose to "update" the disk in the final FIXDISK prompt.

FIXDISK requires 15 minutes to execute and displays each block it reads. If a bad block is found, FIXDISK displays an error message.

FIXDISK completes in 15 minutes, after every sector on disk has been checked. The following prompt displays:

Total of xxx bad blocks found, xxx more this pass.
Want the disk updated? [y,n]

where "xxx" is the number of bad blocks.

Press **Y** to update the disk and mark bad sectors, or press **N** to exit FIXDISK without updating.

Choosing **Y** marks the bad block(s) in files unusable, and stores the current number of bad blocks on the disk. Data in the bad sector is no longer usable, and FIXDISK does not indicate which files may contain a bad sector.

Pressing **N** does not mark bad sectors and does not allow FIXDISK to store the current tally of bad blocks.

FOR

TYPE Internal

PURPOSE Command used in batch and interactive file processing which allows repeated execution of DOS commands.

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

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

REMARKS The FOR command causes DOS to execute <command> repeatedly for each %%<c> or %<c> listed in <set>. <c> can be any character except 0 - 9 (to avoid confusion with the %0 - %9 batch parameters).

<set> is (member [member] [member] ...)

DOS sets %%<c> or %<c> equal to each member of <set>, then executes the <command> for each member. Wild card characters (* and ?) are allowed if only one member is listed for <set> (e.g., *.TXT). <Set> cannot exceed 40 characters, including blanks.

Examples of two batch file commands are shown below:

```
FOR %%f IN ( *.* ) DO DIR %%f
```

The command above results in an individual directory listing of each file on the disk.

```
FOR %%s IN (This is an example.) DO REM %%s
```

The command above results in the following listing:

```
REM This
REM is
REM an
REM example.
```

Only one FOR command can be specified on a single command line in a batch file. Pathnames are not accepted with filenames.

FORMAT

TYPE External

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

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

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

Note: Formatting destroys any data which may previously exist on the disk.

The following switches may be specified:

The **/1** switch formats a single-sided diskette. A single-sided diskette holds 179,712 total bytes after formatting, instead of the 362,496 bytes available after the default FORMAT command.

The **/8** switch formats a disk at eight sectors per track (instead of the default of nine sectors per track). This switch leaves 322,560 bytes available after formatting.

The **/O** switch produces an IBM personal Computer DOS version 1.x compatible disk. The /O switch reconfigures the directory with an OE5 hex byte at the start of each entry so that the disk may be used with 1.x versions of IBM PC DOS, as well as MS-DOS 1.25/2.00 and IBM PC DOS 2.00. This switch noticeably decreases 1.25 and 2.00 performance on disks with few directory entries. The /O switch takes a long time to complete; completion is successful only when the MS-DOS prompt reappears.

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

IO.SYS
MSDOS.SYS
COMMAND.COM

After you type the FORMAT command:

1. The message "Formatting..." is displayed.
2. The message "Format complete" is displayed and you are prompted for the volume label.
3. Bytes of total disk space, bytes used by the system (if any), and bytes available on disk are displayed.
4. The prompt "Format another (Y/N)?" appears.

GOTO

TYPE Internal

PURPOSE Command extension used in batch file processing which transfers control to a specified batch file line.

SYNTAX GOTO <label>

REMARKS GOTO causes commands to be taken from the batch file beginning with the line after the <label> definition. A label in a batch file is a colon (:) followed by a label name. If no label has been defined, the current batch file terminates, displaying the message **Label not found.** GOTO causes the command(s) that immediately follow <label> to be executed.

For example:

```
:loop
REM looping...
GOTO loop
```

produces an infinite sequence of messages:

```
REM looping...
```

Starting a line in a batch file with ":" causes the line to be ignored by batch processing. Therefore, although the characters following ":" define a label, they may also be used to insert comments into your file. For example, using the batch setup above, you could type:

```
:loop This is an infinite loop.
```

or

```
:loop
:This is an infinite loop.
```

None of these statements are displayed on your screen when you run your batch file. They appear only in a listing of your batch file.

HDFORMAT

TYPE External

PURPOSE HDFORMAT formats hard disks for use with MS-DOS.

SYNTAX HDFORMAT C:

REMARKS Attache allows the drive reference **C:** only. If any other drive is referenced or the disk is not ready, HDFORMAT issues the error message **You MUST specify C: to format!**

When Drive C is selected, HDFORMAT displays the following message:

```
CAUTION - RUNNING THIS PROGRAM
WILL PERMANENTLY DESTROY ALL DATA
ON THE DISK!  TYPE ^C TO ABORT
```

HDFORMAT destroys all data on the disk and should only be executed during the initial installation of the disk. Press **RETURN** to begin formatting the hard disk. If you erroneously enter HDFORMAT, you can press **CTRL C** to exit the program and return to DOS.

Formatting requires three minutes, during which time Attache displays the following prompt:

```
Wait, formatting ...
```

HDFORMAT verifies the formatting data on all tracks and displays each track being verified, stopping on the final track, 305. This executes in less than 30 seconds. The following message is then displayed:

```
Volume label (11 characters, ENTER for none)?
```

You may either enter a volume label of up to 11 characters for your disk, or press **RETURN** for no label. The following message is displayed:

```
9689088 bytes total disk space
9689088 bytes available on disk
A>
```

The disk is configured as Drive C. At each system boot, Drive A is the current logged drive.

IF

TYPE Internal

PURPOSE Command extension used in batch file processing to allow conditional execution of commands.

SYNTAX IF [NOT] <condition> <command>

REMARKS The IF statement allows conditional execution of commands. When the <condition> is true, the <command> is executed. Otherwise, the <command> is ignored. The parameter <condition> is one of the following:

ERRORLEVEL <number>
True only if the previous program executed by COMMAND had an exit code of <number> or higher, where <number> is a binary value.

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

EXIST <filename>
True only if <filename> exists. Multiple filenames, pathnames, and directories cannot be used for <filename>, but wild cards are acceptable.

Examples:

```
IF NOT EXIST ACCTS.TXT ECHO Can't find file
```

This batch file statement displays "Can't find file" if ACCTS.TXT is not located in the current directory. Because pathnames are not allowed in an IF statement, a preceding CHDIR statement may be required in the batch file. (Batch files always return to their original directory upon termination, regardless of any CHDIR statements they might contain.)

```
IF ERRORLEVEL 1 LINK,,;
```

The batch file containing this command can run after a routine which sets error levels. Any condition except Errorlevel 1 causes LINK to execute.

```
IF %1==ACCTS GOTO ACCOUNTS
```

Executing this batch file with ACCTS given as the value to substitute for %1 causes the commands after the label :ACCOUNTS to execute.

LINK

TYPE External

PURPOSE Compile and link programs.

SYNTAX LINK [<filenames>][/switches] or LINK @<filespec>

REMARKS The MS-DOS linker program (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; and produces a listing that shows both the resolution of external references and error messages.

MS-LINK combines several object files into one relocatable load module, or Run file (called an .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 also produces a List file that shows external references resolved, and displays any error messages. MS-LINK uses available memory if possible. When available memory is exhausted, MS-LINK creates a temporary disk file named VM.TMP.

MS-LINK may be started in three ways: you may type commands in response to individual prompts, you may type all commands on the line used to start MS-LINK, or you may create a response file that contains all the necessary commands and tell MS-LINK where that file is when you start MS-LINK.

MS-LINK provides three command characters: the plus sign (+), semicolon (;), and **CTRL C**.

Details on running MS-LINK, the command characters, switches, and files are in the Programmer's Reference Guide.

MAKEDB

TYPE External

PURPOSE Calculates the decimal value of each byte of a binary input file. This is useful for entering data blocks or BASIC data statements.

SYNTAX MAKEDB <file>

REMARKS MAKEDB calculates and displays the decimal equivalent of hexadecimal bytes. This may be used for entering large arbitrary blocks of data to an assembler or for BASIC data statements.

MAKEDB keeps all lines under 47 characters in length, which fits on 80 column paper when used with MASM. Output is sent to the console, so you'll need to use redirection to save the output.

For example, if your binary input file contains:

```
0000:  00 10 20 c0 ff 00 10 20 c0 ff 00 10 20 c0 ff
```

the output looks like:

```
db      0,16,32,192,255,0,16,32,192
db      255,0,16,32,192,255,0,16,32
db      192,255
```

MAKEDB can also calculate the ASCII decimal equivalent of ASCII files. For example, if your ASCII file contains:

```
This is an example.
```

the output from MAKEDB would look like:

```
db      84,104,105,115,32,105,115,32
db      97,110,32,101,120,97,109,112
db      108,101,46
```

MKDIR or MD - Make Directory

TYPE Internal

PURPOSE Makes a new directory.

SYNTAX MKDIR [d:]<pathname>

REMARKS 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 as in the following example:

```
MKDIR \USER
```

This creates a subdirectory \USER in your root directory. While in the root directory, you can create a directory named JOE under \USER:

```
MKDIR \USER\JOE
```

Using this same example, if you are in USER, you could create the same directory by typing:

```
MKDIR JOE
```

The first "\" tells DOS to begin its directory search with the root directory. If the command does not contain a leading "\", DOS begins at the current directory.

Note: The maximum number of characters you can type for a single path from the root directory to any subdirectory must not exceed 63 characters, including imbedded backslashes.

MORE

TYPE External

PURPOSE Sends output to console one screen at a time.

SYNTAX [d:]MORE

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

Pressing **RETURN** displays another screen of information. This process continues until all the input data is read.

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

```
TYPE MYFILES.COM | MORE
```

MS-DOS displays the file MYFILES.COM (on the default drive) one screen at a time. If the MORE command is not in your default disk drive, you may add the drive specification (e.g., B:MORE).

MOVAFILE

TYPE External

PURPOSE MOVAFILE may be used to copy modified files to hard disk or diskette. It replaces altered files on the destination drive automatically.

SYNTAX MOVAFILE [s:][filename][.ext][d:][filename][.ext] [-A]

where -A specifies copy only files that have the Archive attribute set,
s: is the source drive reference,
d: is the destination drive reference,
filename is the name of the file or group of files,
.ext is the filename extension.

An asterisk (*) can be used as a wild card for the filename or extension if desired.

MOVAFILE also checks for invalid filenames, such as null strings, and displays the erroneous filename without aborting.

PATH

TYPE Internal

PURPOSE Sets a command path.

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

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

To tell MS-DOS to search the root directory of Drive A for external commands, type:

```
PATH A:\
```

If your external commands are stored in a subdirectory with the pathname MASTER, type:

```
PATH \MASTER
```

To tell MS-DOS to search your JOE directory in the path \MASTER\USER\JOE for external commands, type:

```
PATH \MASTER\USER\JOE
```

MS-DOS searches the JOE directory for the commands you enter until you set another path or shut down MS-DOS. Note that in all cases the directory which will be searched is the last directory specified.

If a command is not found in the specified path, DOS displays the message **bad command or filename.**

To search more than one path, specify several pathnames separated by semicolons. For example, to search the JOE, SUE, and DEV subdirectories, type:

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

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

The command **PATH** with no options displays the current path or the message "No Path". If you specify **PATH ;**, MS-DOS sets the NUL (no extended search) path, so only the current directory is searched for external commands.

The PATH command only needs to be entered once per DOS session (unless you wish to change it).

A PATH command may be included in your AUTOEXEC.BAT file to automatically select the desired path when you boot MS-DOS. For example, if you always wish to search the root directory on Drive A for external commands, you may enter **PATH A:** in your AUTOEXEC.BAT file.

PAUSE

TYPE Internal

PURPOSE Suspends execution of the batch file.

SYNTAX PAUSE [comment]

REMARKS 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:

Strike a key when ready . . .

If you press **CTRL C**, another prompt is displayed:

Abort batch job (Y/N)?

If you type **Y** in response to this prompt, execution of the remainder of the batch command file is aborted and control returns to the 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.

[Comment] is optional and may be entered on the same line as PAUSE. A comment can contain up to 121 characters and may be used to prompt the user of the batch file with some meaningful message when the batch file pauses. For example, you may want to change disks in one of the drives. An optional prompt message may be given in such cases. The comment prompt is displayed before the "Strike a key" message.

PRINT

TYPE External

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

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

REMARKS Up to ten files can be queued for printing at one time. Using the PRINT command increases the resident portion of DOS by more than 3K bytes.

Only files located in the current directory can be placed into the print queue, however, you can change directories as soon as the file is queued.

The following switches are provided with this command:

/T TERMINATE: This switch deletes all files in the print queue (i.e. the files waiting to be printed). Current printing stops, and a cancellation message prints.

/C CANCEL: This switch turns on cancel mode. The preceding filespec and all following filespecs are canceled from the print queue until you type a **/P** switch.

/P PRINT: This switch turns on print mode. The preceding filespec and all following filespecs are added to the print queue until you issue a **/C** switch. (/P is the default switch.)

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

Examples:

PRINT /T -- Empties the print queue.

PRINT /T *.ASM -- Empties the print queue and queues all .ASM files on the default drive.

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

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

Error Messages

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

Name of list device [PRN:] -- This prompt appears when PRINT is run the first time. Any current device may be specified which then becomes the PRINT output device. Pressing **RETURN** results in the device PRN being used.

List output is not assigned to a device -- This message is displayed if the "Name of list device" specified to the above prompt is invalid. Subsequent attempts return the same message until a valid device is specified.

PRINT queue is full -- If you attempt to put more than 10 files in the queue, this message appears on the console.

PRINT queue is empty -- There are no files in the print queue.

No files match d:xxxxxxxx.xxx -- A filespec was given for files to add to the queue, but no files match the specification. **Note:** If no files in the queue match the canceled filespec, no error message appears.

Drive not ready -- If this message occurs when PRINT attempts a disk access, PRINT keeps trying until the drive is ready. Any other error causes the current file to be canceled. An error message appears on your printer in such a case.

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

PROMPT

TYPE Internal

PURPOSE Changes the MS-DOS command prompt.

SYNTAX PROMPT [<prompt-text>]

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

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

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

The **PROMPT** command with no options restores the normal MS-DOS prompt (e.g., A>).

PROMPT Time = t_Date = $d sets a two-line prompt which prints:

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

You can use escape sequences in your prompts. For example, **PROMPT $e[7m$n:$e[m** sets the prompts in inverse video mode and returns to video mode for other text.

Prompts can be executed automatically at boot by entering your PROMPT command into an AUTOEXEC.BAT file.

RECOVER

TYPE External

PURPOSE Recovers a file that contains bad sectors.

SYNTAX RECOVER <filespec>

REMARKS If a sector on a disk is bad, you can recover the file containing that sector (the bad sector is deleted). If the bad sector was in the directory, use CHKDSK.

To recover a particular file, type:

RECOVER <filename>

This causes MS-DOS to read the file sector by sector and to skip the bad sector(s). When MS-DOS finds the bad sector(s), the sector(s) are marked and MS-DOS no longer allocates your data to that sector. The sector is permanently marked unusable.

The size of the recovered file is a multiple of the DOS allocation unit size. The recovered file is therefore usually larger than the original file size. Text files often require editing to remove unwanted data at the end of the recovered file.

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

Note: If RECOVER is used on an entire disk, all files are renamed FILEnnnn.REC (where nnnn is a sequential number) and all subdirectories are deleted. If a bad sector is located in the disk directory, use the command CHKDSK to repair the directory.

REM - REMARK

TYPE Internal

PURPOSE Displays remarks written within a batch file.

SYNTAX REM <comment>

REMARKS When batch processing reaches the REM command, it displays the <comment>. Comments can be any string of characters up to 123 characters long. REM with no comment may be used to provide spacing within a file.

REN or RENAME

TYPE Internal

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

SYNTAX REN <filespec> <filename>

REMARKS The first option is the file to be renamed, and the second option is the new filename. The first <filespec> must be given a drive designation if the disk resides in a drive other than the default drive. Any drive designation for the second option (filename) is ignored. The file remains on the disk where it currently resides.

The wild card characters question mark (?) or asterisk (*) 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 are not changed.

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

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

In the next example, REN renames the file ABODE on Drive B to BBODE:

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

The file remains on Drive B.

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

RMDIR or RD - Remove Directory

TYPE Internal

PURPOSE Removes a directory from the specified disk.

SYNTAX RMDIR <pathname>

REMARKS This command removes a directory which is empty except for the . and .. shorthand symbols.

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

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

The directory is deleted from the directory structure.

SET

TYPE Internal

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

SYNTAX SET [<string=string>]

REMARKS This command sets values to be used by your application programs. An application program can check all values that have been set with the SET command by issuing **SET** with no options. For example, **SET TTY=VT52** sets your TTY value to VT52 until you change it with another SET command. Typing **SET** without parameters shows the current setting(s) of SET.

The SET command can also be used in batch processing to allow you to define your replaceable parameters with names instead of numbers. If your batch file contains the statement "LINK %FILE%", the SET command can specify the name that MS-DOS uses for that variable. For example, to replace the %FILE% parameter with the filename DOMORE:

```
SET FILE=DOMORE
```

Therefore, you do not need to edit each batch file to change the replaceable parameter names.

Note: When you use text (instead of numbers) as replaceable parameters, the name must be ended by a percent sign.

SHIFT

TYPE Internal

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

SYNTAX SHIFT

REMARKS 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, suppose a batch file named FILE.BAT contains the following commands:

```
echo off
echo %0 %1 %2 %3 %4 %5 %6 %7 %8 %9
shift
echo %0 %1 %2 %3 %4 %5 %6 %7 %8 %9
shift
echo %0 %1 %2 %3 %4 %5 %6 %7 %8 %9
```

Typing the command **FILE 1 2 3 4 5 6 7 8 9 A B** would produce the following lines:

```
file 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 a
2 3 4 5 6 7 8 9 a b
```

SIZE

TYPE External

PURPOSE Displays the filename and size of each file, and calculates the total size of all files.

SYNTAX SIZE [<filespec>]

REMARKS This command may be used before copying a large number of files from a hard disk to diskette(s) to determine whether or not the specified file(s) will fit. You can specify a pathname with the filename if desired. If no name is entered, the SIZE parameter defaults to *.*.

For example, typing **SIZE *.BAT** may produce the following listing:

```
file.bat      75
rem.bat       40
run.bat      163
3 files, total of 278 bytes
```

Note: SIZE lists the total byte count, not the amount of disk space to be allocated. Because MS-DOS allocates in clusters, it is possible for SIZE to indicate that all files will fit on the destination diskette, and have the disk fill up before the operation is complete.

SORT

TYPE External

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

SYNTAX SORT [/R] [/+n]

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

/R reverse the sort (i.e., sort from Z to A).

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

Examples:

The following command reads the file UNSORT.TXT, reverses the sort, and writes the output to a file named SORT.TXT:

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

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

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

Additionally, the MORE filter can be used to allow you to read the sorted directory one screen at a time. For example:

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

SYS - System

TYPE External

PURPOSE Transfers the MS-DOS system files from the disk in the default drive to the disk in the drive specified by d:.

SYNTAX SYS <d:>

REMARKS SYS is normally used to update the system or to place the system on a blank, formatted disk. An entry for the drive <d:> is required.

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

The transferred files are copied in the following order:

IO.SYS
MSDOS.SYS

IO.SYS and MSDOS.SYS are both hidden files that do not appear when the DIR command is executed.

COMMAND.COM (the command processor) is not transferred. You must use the COPY command to transfer COMMAND.COM. If you do not copy COMMAND.COM and attempt to boot the system, you receive the message **Bad or missing Command interpreter.**

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

No room for system on destination disk -- There is not enough room on the destination disk for the IO.SYS and MSDOS.SYS files.

Incompatible system size -- The system files IO.SYS and MSDOS.SYS do not take up the same amount of space on the destination disk as the new system needs.

TIME

TYPE Internal

PURPOSE Displays and sets the time.

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

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

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

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

```
TIME 8:20
```

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

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

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

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

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

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

TREE

TYPE External

PURPOSE TREE displays subdirectories on the disk from the current directory down and, optionally, files within the subdirectories.

SYNTAX TREE [<filename>] [A]

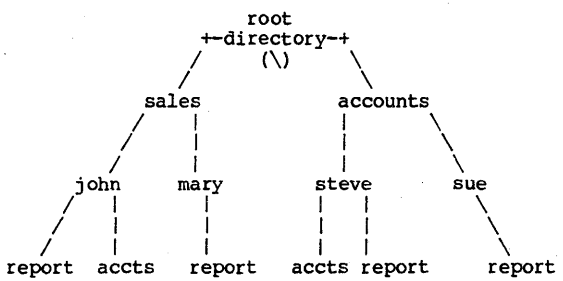
REMARKS TREE can display the subdirectory structure, and specific files within the structure. You can use TREE at the root to find a file within any directory.

TREE allows you to list only files that have been modified. If you specify the letter "A" after a filename, it lists only files that have the Archive attribute set. Whenever you modify a file in any way, DOS sets the "A" attribute to mark it as "changed."

If you have copied your files and wish to remove the "A" attribute, you can run the FILETYPE program described earlier in this chapter. If you backup your files using the ARCHIVE command, the "A" attribute is cleared automatically.

Examples

Assume your disk has the following directories:



When you are in the root directory (\), running TREE displays:

```
SALES
  JOHN
    REPORT
    ACCTS
  MARY
    REPORT
ACCOUNTS
  STEVE
    ACCTS
    REPORT
  SUE
    REPORT
```

If you are in the SALES directory and you run TREE, it lists:

```
JOHN
  REPORT
  ACCTS
MARY
  REPORT
```

TREE works from the current directory down to the lowest level it finds. It never looks "up" the tree.

TREE can display specified files. If you know there's a file called MONEY.$$$ somewhere on the disk, instead of doing a CHDIR then a DIR in every directory, you could go to the directory root and type:

```
TREE money.$$$
```

The following report is then displayed:

```
SALES
  JOHN
    REPORT
    ACCTS
  MARY
    REPORT
      money.$$$
ACCOUNTS
  STEVE
    ACCTS
    REPORT
  SUE
    REPORT
```

If the file is found, it is listed in lowercase to distinguish it from directory names, which are always in uppercase. You can use wild cards in the filename.

TYPE

TYPE Internal

PURPOSE Displays the contents of the file on the console screen.

SYNTAX TYPE <filespec>

REMARKS Use this command to examine a file without modifying it. The only formatting performed by TYPE is that tabs are expanded to the tab stops, located at every eighth column. Note that a display of binary files may be unreadable, as control characters (such as **CTRL Z**) are sent to your display, including bells, form feeds, and escape sequences.

Note: WordStar files do not display properly with the TYPE command. These files must be viewed within WordStar.

Wild cards are not allowed in the filespec. If a wild card character is used, the message **File not found** displays.

VER - Version

TYPE Internal

PURPOSE Prints MS-DOS Version number.

SYNTAX VER

REMARKS If you want to know what version of MS-DOS you are using, type:

```
VER
```

The version number is displayed on your screen. For example:

```
MS-DOS Version 2.0
```

VERIFY

TYPE Internal

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

SYNTAX VERIFY [ON|OFF]

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

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

Typing **VERIFY** with no options displays the current setting of VERIFY.

VOL - Volume

TYPE Internal

PURPOSE Displays disk volume label, if one exists.

SYNTAX VOL [d:]

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

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

Volume in drive x has no label

DOS Guide

DOS Editing and Function Keys

DOS Editing and Function Keys

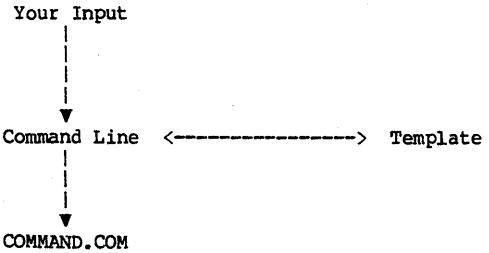
Overview

Special editing keys allow you to re-enter or modify MS-DOS commands. Using these keys, you do not have to repeatedly type commands because the last command line typed is automatically placed in a special storage area called a template.

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

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

The relationship between the command line and the template is illustrated below:



As shown above, you type a command to MS-DOS on the command line. When you press the **RETURN** key, the command is automatically sent to the command processor (COMMAND.COM) for execution. At the same time, a copy of this command is sent to the template. You can now recall the command or modify it with MS-DOS special editing keys.

MS-DOS editing keys for Attache are listed in the table below:

MS-DOS Editing Keys

Function	Description	Attache	IBM PC
<COPY1>	Copies one character from the template to the command line	**CTRL 1**	F1
<COPYUP>	Copies characters up to the specified character in the template and puts them on the new command line	**CTRL 2**	F2
<COPYALL>	Copies all the remaining characters in the template to the command line	**CTRL 3**	F3
<SKIP1>	Skips over (does not copy) a character in the template	**DEL**	DEL
<SKIPUP>	Skips over (does not copy) characters in the template up to the specified character	**CTRL 4**	F4
<VOID>	Voids current input; leaves the template unchanged	**CTRL X**	ESC
<INSERT>	Enters and exits insert mode	**CTRL DEL**	INS
<REPLACE>	Exits insert mode	**CTRL DEL**	INS
<NEWLINE>	Makes the new line the new template	**CTRL 5**	F5

Examples:

If you type the following command:

DIR PROG.COM

MS-DOS displays information about the file PROG.COM on your screen. The command line is also saved in the template. To repeat the command, just press **CTRL** 3 and **RETURN**. Pressing **CTRL** 3 causes the contents of the template to be copied to the command line, and pressing **RETURN** causes the command line to be sent to the command processor for execution.

If you want to display information about a file named PROG.ASM, you can use the contents of the template by pressing **CTRL** 2 and then typing C. This copies all the characters from the template to the command line, up to but not including C. MS-DOS displays:

```
DIR PROG._
```

The underline is your cursor. Now type:

```
ASM
```

The result is:

```
DIR PROG.ASM_
```

The command line DIR PROG.ASM is now in the template and ready to be sent to the command processor for execution. To execute it, press **RETURN**.

Now assume that you want to execute the following command:

```
TYPE PROG.ASM
```

To do this, type the word **TYPE**, press **CTRL DEL** and the space bar, then press **CTRL** 3 and **RETURN**.

As you type, the characters are entered directly into the command line and overwrite corresponding characters in the template. Thus, the characters "TYPE" replace the characters "DIR " in the template. This automatic replacement feature is turned off when you press **CTRL DEL**.

To insert a space between "TYPE" and "PROG.ASM", press **CTRL DEL** and the space bar. Finally, to copy the rest of the template to the command line, press **CTRL** 3 and **RETURN**. The command TYPE PROG.ASM is now processed by MS-DOS, and the template becomes TYPE PROG.ASM.

If you had misspelled "TYPE" as "BYTE", a command error would have occurred when you pressed **RETURN**. Instead of throwing away the whole command, you can save a misspelled line before you press **RETURN** by creating a new template with **CTRL 5**.

Pressing **CTRL 5** allows you to edit a line without executing the command. An @ sign appears after the line to signal that a new template has been created:

```
BYTE PROG.ASM@
```

You can edit this erroneous command by typing **T** to replace the "B", then pressing **CTRL 1** to copy the "Y." Pressing **CTRL 3** copies the rest of the template. The resulting command line is then the command that you want:

```
TYPE PROG.ASM
```

As an alternative, you can modify the BYTE PROG.ASM template by using **DEL** and **CTRL DEL** as shown below. Note the affect the keys typed on the left have on the command line:

DEL	_	Skips over 1st template character
DEL	_	Skips over 2nd template character
CTRL 1	T	Copies 3rd template character
CTRL DEL	TYP	Inserts two characters
CTRL 3	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, pressing **CTRL 4** deletes characters in the template, up to but not including a given character, and does not alter the command line.

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.

Control Character Functions

Control character functions give you greater control over your display screen and the execution of MS-DOS commands. Control character functions are activated by holding down the **CTRL** key and while you press the required character key.

Control Character Functions

Control Character	Function
CTRL C	Aborts current command.
CTRL H	Removes last character from command line and erases it from the terminal screen.
CTRL J or LINE FEED	Inserts a line-feed carriage-return, but does not execute the command. This may be used to extend the current logical line beyond the physical limits of one terminal screen.
CTRL N or CTRL P	Toggles terminal output to the printer; echos screen display.
CTRL S	Suspends output display on terminal screen. Press any key to resume.
CTRL X	Cancels the current line. The command is not executed and a backslash (\), carriage return, and line feed appear on the screen. The template is not affected.
CTRL Z	Puts a CTRL Z (1AH) end-of-file character in the new template.

DOS Guide

EDLIN - The Line Editor

EDLIN - The Line Editor

Overview

This chapter describes how to use EDLIN, the line editor program. You can use EDLIN to create, change, and display either source program files or text files. 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 in one or more lines.

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

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

How to Start EDLIN

To start EDLIN, type:

```
EDLIN <filespec> [/B]
```

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

```
New file
*_
```

The prompt for EDLIN is an asterisk (*).

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

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

```
End of input file
*
```

You can then edit the file using EDLIN editing commands.

If you have already created a file, the optional /B switch may be used. The /B switch ignores any CTRL Z characters in the file, which allows the entire file to be displayed.

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

To edit the remainder of the file, save some of the edited lines in memory back to disk. EDLIN then can append the unedited lines on disk to the lines remaining in memory. See the Write and Append commands in this chapter for this procedure.

When you complete the editing session, save the original and updated (new) files using the End command. The End command is discussed below in the section "EDLIN Commands." The original file is renamed with the extension .BAK and the new file with the filename and extension specified in the EDLIN command. The original .BAK file is not erased until the end of the editing session or until disk space is needed by 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 need to edit such a file, first rename the file with another extension (using the MS-DOS RENAME command discussed in Chapter 5), then start EDLIN and specify the new <filespec>.

Special Editing Keys

The special editing keys discussed in Chapter 6 can be used to edit your text files. The special editing keys use a DOS convention called a "template", which is an area in memory that stores the command line. Editing keys can use the template to restore all or some of the characters from the previous command line. These keys are discussed in detail in this section and are illustrated in the table on the next page.

The following chart shows the function each key performs, the DOS key description, the IBM Personal Computer keys which perform that function, and the keys to press on an Attache 8:16.

Special Editing Keys

Function	DOS Key	IBM Key	8:16 Key	Description
Copy one character	<COPY1>	F1	CTRL 1	Copies one character from the template to the new line.
Copy up to character	<COPYUP>	F2	CTRL 2	Copies all characters from the template to the new line, up to the specified character.
Copy template	<COPYALL>	F3	CTRL 3	Copies all remaining characters in the template to the screen.
Skip one character	<SKIP1>	DEL	DEL	Does not copy (skips over) a character.
Skip up to character	<SKIPUP>	F4	CTRL 4	Does not copy (skips over) the characters in the template, up to the specified character.
Quit input	<VOID>	ESC	CTRL X	Voids the current input; leaves the template unchanged.
Insert mode	<INSERT>	INS	CTRL DEL	Enters/exits insert mode.
Replace	<REPLACE>	INS	CTRL DEL	If in insert mode, turns insert off to replace. Replace is default mode.
New template	<NEWLINE>	F5	CTRL 5	Makes the new line the new template.

The next section contains several examples which use a sample file to demonstrate these keys. To create this file, type **EDLIN SAMPLE.TXT.** When the **NEW FILE** * prompt is displayed, type **I** and press **RETURN** to enter insert mode. The prompt **1*** is displayed, indicating that you can insert data into the first line of your file.

The examples in the following section use a one-line file. Enter the following text, then press **RETURN**:

```
This is a sample file.
```

Press the **CTRL Z** keys, and then **RETURN**. This returns you to EDLIN command mode. Next, save your file and exit EDLIN by pressing **E** and **RETURN**.

To edit SAMPLE.TXT with the special editing keys, type **EDLIN SAMPLE.TXT**. The following prompt displays:

```
End of input file
*
```

Type **1** and press **RETURN** to edit the first line of text, using the instructions which follow.

KEY **<COPY1> - CTRL 1**

PURPOSE Copies one character of the template to a command line.

REMARKS Pressing **CTRL 1** copies one character from the template to the command line. When **CTRL 1** is pressed, one character is inserted in the command line and insert mode is automatically turned off. For example, assume that the screen 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 **CTRL 1** copies the first character (T) to the second of the two lines displayed:

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

Each time **CTRL 1** is pressed, one more character appears:

```
CTRL 1 1:*Th_
CTRL 1 1:*Thi_
CTRL 1 1:*This_
```

KEY <COPYUP> - CTRL 2

PURPOSE Copies all characters up to a given character.

REMARKS Pressing **CTRL 2** copies all characters up to a given character from the template to the command line. The given character is the next character typed after **CTRL 2**; it is not copied or displayed on the screen.

Pressing **CTRL 2** 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 **CTRL 2** also automatically turns off insert mode.

For example, assume that the screen 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 **CTRL 2** copies all characters up to the character specified immediately after the **CTRL 2** keys.

```
                1:*This is a sample file.
CTRL 2 p        1:*This is a sam_
```

KEY <COPYALL> - CTRL 3

PURPOSE Copies template to command line.

REMARKS Pressing **CTRL 3** copies all remaining characters from the template to the command line. Regardless of the cursor position at the time **CTRL 3** is pressed, the rest of the line appears and the cursor is positioned after the last character on the line. For example, the screen shows:

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

Pressing **CTRL 3** copies all characters from the template (upper line) to the line with the cursor (lower line):

```
                1:*This is a sample file.  (template)
CTRL 3          1:*This is a sample file._ (command line)
```

Insert mode is automatically turned off.

KEY <SKIP1> - DEL

PURPOSE Skips over one character in the template.

REMARKS 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 the **CTRL 1** function, except that **DEL** skips a character in the template instead of copying it to the command line. For example, assume that the screen 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 **CTRL 3**, which moves the cursor beyond the last character of the line.

```
                1:*This is a sample file.
DEL             1:*_
CTRL 3          1:*his is a sample file._
```

KEY <SKIPUP> - CTRL 4

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

REMARKS Pressing **CTRL 4** skips over all characters up to a given character in the template. This character is not shown on the screen. If the template does not contain the specified character, nothing is skipped over. The action of **CTRL 4** is similar to that of **CTRL 2**, except that **CTRL 4** skips over characters in the template instead of copying them to the command line. For example, assume that the screen 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 **CTRL 4** skips over all the characters in the template up to the character pressed after the **CTRL 4**:

```
                1:*This is a sample file.
CTRL 4 p        1:*_
```

The cursor position does not change.

To see how much of the line has been skipped over, press **CTRL 3** to copy the template. This moves the cursor beyond the last character of the line:

```
                1:*This is a sample file.
CTRL 4 p        1:*_
CTRL 3          1:*ple file._
```

KEY **<VOID> - CTRL X**

PURPOSE Quits input and empties the command line.

REMARKS Pressing **CTRL X** empties the command line, but leaves the template unchanged. **CTRL X** also prints a backslash (\), carriage return, and line feed, turns insert mode off, and positions the cursor at the beginning of the line. Pressing **CTRL 3** copies the template to the command line and the command line appears as it was before **CTRL X** was pressed. For example, assume that the screen 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. To replace the line with "Sample File," type that phrase as shown:

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

To cancel the line you just entered and keep the phrase "This is a sample file.", press **CTRL X.** A backslash appears on the "Sample File" line as shown to tell you it has been canceled:

```
                1:*This is a sample file.
CTRL X          1:*Sample File\
                1:_
```

Press **RETURN** to keep the original line or to perform any other editing function. If **CTRL 3** is pressed, the original template is copied to the command line:

```
CTRL 3      1: This is a sample file._
```

KEY **<INSERT> - CTRL DEL**

PURPOSE Enters/exits insert mode.

REMARKS Pressing **CTRL DEL** causes EDLIN to enter and exit insert mode. The current cursor position in the template is not changed. The cursor moves as each character is inserted; when you have finished inserting characters, the cursor will be 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. For example, assume that the screen 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. For example, press **CTRL 2** and **f**:

```
                1:*This is a sample file.
CTRL 2 f        1:*This is a sample _
```

Now press **CTRL DEL** and insert the word "edit" and a space:

```
                      1:*This is a sample file.
CTRL 2 f              1:*This is a sample _
CTRL DEL edit 1:*This is a sample edit _
```

Press **CTRL 3** to copy the rest of the template to the line:

```
                      1:*This is a sample edit _
CTRL 3                1:*This is a sample edit file._
```

If you press **RETURN** instead of **CTRL 3**, the remainder of the template is truncated and the command line ends after the inserted phrase:

```
CTRL DEL edit RETURN   1:*This is a sample edit _
```

To exit insert mode, press **CTRL DEL** again.

KEY **<REPLACE> - CTRL DEL**

PURPOSE Enters replace mode.

REMARKS When you first start to edit a line, replace mode is in effect (**CTRL DEL** does not have to be pressed). All the characters you type overstrike and replace characters in the template. If you are in insert mode, pressing **CTRL DEL** causes EDLIN to exit insert mode and enter replace mode (**CTRL DEL** is pressed twice; once to enter insert mode, and again to enter replace mode). If **RETURN** is pressed, the remainder of the template is deleted. For example, assume that the screen 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 press **CTRL 2** m, **CTRL DEL** lary, **CTRL DEL** tax, and **CTRL 3** as shown:

```
                  1:*This is a sample file.
CTRL 2 m          1:*This is a sa_
CTRL DEL lary     1:*This is a salary_
CTRL DEL tax      1:*This is a salary tax_
CTRL 3            1:*This is a salary tax file._
```

You inserted **lary** and replaced **mple** with **tax.** If you type characters that extend beyond the length of the template, the remaining characters in the template automatically are appended when you press **CTRL 3.**

KEY **<NEWLINE> - CTRL 5**

PURPOSE Creates a new template.

REMARKS Pressing **CTRL 5** copies the current command line to the template and deletes the contents of the old template. Pressing **CTRL 5** produces an @ ("at" sign), a carriage return, and a line feed. The command line is emptied and insert mode is turned off.

Note: **CTRL 5** performs the same function as the **CTRL X** key except that the template is changed and an @ ("at" sign) is printed instead of a backslash (\). For example, assume that the screen 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 you enter **CTRL 2 m**, **CTRL DEL lary**, **CTRL DEL tax**, and **CTRL 3** as shown:

```
                  1:*This is a sample file.
CTRL 2 m          1:*This is a sa_
CTRL DEL lary     1:*This is a salary_
CTRL DEL tax      1:*This is a salary tax_
CTRL 3            1:*This is a salary tax file._
```

If you want this line to be the new template, press **CTRL 5**:

```
CTRL 5          1:*This is a salary tax file.@
```

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

Command Information

EDLIN commands perform editing functions on lines of text. You should read the following information before using EDLIN commands.

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

2. You can reference line numbers relative to the current line (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**

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

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

   For example, the following command line edits line 15, then displays lines 10 through 20 on the screen:

   15;-5,+5L

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

   SThis string CTRL Z -5,+5L

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

5. It is possible to insert a control character (such as **CTRL C**) into the text by preceding it with the quote character **CTRL V** while in insert mode. **CTRL V** tells MS-DOS to recognize the next capital letter typed as a control character. You also may include a control character in any of the string arguments of Search or Replace by using the special quote character **CTRL V.** For example:

   S CTRL-V Z finds the first occurrence of CTRL Z in a file.

   R CTRL-V Z CTRL-Z name replaces all occurrences of CTRL Z in a file with "name".

   S CTRL-V C CTRL-Z bar replaces all occurrences of CTRL C with "bar".

   It is also possible to insert **CTRL V** into the text by typing **CTRL-V V.**

6. The **CTRL Z** character ordinarily signals "end of file" to EDLIN. If you have **CTRL Z** characters elsewhere in your file, you must tell EDLIN that these control characters do not mean end-of-file by using the /B switch when you first call up the file under EDLIN. The /B switch causes EDLIN to ignore any **CTRL Z** characters so you may view the entire file.

Command Options

Several EDLIN commands accept one or more options. The effect of a command option varies depending on the command with which 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, options, and the command. <line> may be specified in one of the following ways:

- Number — Any number less than 65534. If a number larger than the largest existing line number is specified, <line> means the line after the last line number.
- Period — (.) A period specified for <line> means the current line number. The current line is the last line edited and is not necessarily the last line 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. Specifying # for <line> has the same effect as specifying a number larger than the last line number.
- RETURN — A carriage return entered without any of the <line> specifiers listed above directs EDLIN to use the default value for the command.
- Question Mark — (?) The question mark option may be used only with the Replace and Search commands and directs EDLIN to ask you if the correct string has been found. EDLIN waits for either a **Y** or **RETURN** for a "yes" response, or for any other key for a "no" response.

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

EDLIN Commands

EDLIN commands are summarized in the following table. They are described in further detail below in this chapter.

EDLIN Commands

Command	Purpose
A	Appends lines on disk to file in memory
C	Copies lines
D	Deletes lines
<line>	Edits line number
E	Ends editing session
I	Inserts text
L	Lists text
M	Moves lines
P	Pages through a file
Q	Quits editing session
R	Replaces text phrase
S	Searches text for phrase
T	Transfers disk file into text file
W	Writes lines in memory to disk

NAME **Append**

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

SYNTAX [<n>]A

REMARKS This command is meaningful only if the file being edited is too large to fit into memory. As many lines as possible are read into memory for editing when you start EDLIN. To edit the remainder of the file that does not fit into memory, lines that have already been edited must be written to disk using the Write command. You then can load unedited lines from disk into memory with the Append command. See the Write command later in this chapter for more information.

Notes:

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

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

NAME Copy

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

SYNTAX <lstline>,<endline>,<toline>,[<count>]C

REMARKS If you do not specify a number in <count>, EDLIN copies the lines one time. <lstline> and <endline> define the range of lines you wish to copy. <Toline> specifies the line which will contain <lstline>. The file is renumbered automatically after the copy. The line numbers must not overlap or you receive an "Entry error" message. For example, typing **3,20,15C** would result in an error message.

For 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 to copy text.
```

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

```
1,4,5C
```

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 to copy text.
5: This is a sample file
6: used to show copying lines.
7: See what happens when you use
8: the Copy command to copy text.
```

EDLIN inserts copied text immediately before the text currently at <toline>.

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 to copy text.
 5: You can also use Copy
 6: to copy lines of text
 7: to the middle of your file.
 8: End of sample file.
```

The command 3,4,7C 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 to copy text.
 5: You can also use Copy
 6: to copy lines of text
 7: to the middle of your file.
 8: See what happens when you use
 9: the Copy command to copy text.
10: End of sample file.
```

NAME **Delete**

PURPOSE Deletes a specified range of lines in a file.

SYNTAX [<lstline>][,<endline>]D

REMARKS If <lstline> is omitted, the Delete option defaults to the current line (the line with the asterisk next to the line number). If <endline> is omitted, only <lstline> is deleted. When lines have been deleted, the line immediately after the deleted section becomes the current line and has the same line number that <lstline> had.

For 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: Use Delete and Insert
  ...
25: (the D and I commands)
26: to edit the text
27:*in your file.
```

To delete multiple lines, type <lstline>,<endline>D:

```
4,24D
```

The result is:

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

To delete a single line, type:

```
5D
```

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: Use Delete and Insert
4: (the D and I commands)
5:*in your file.
```

Notice that lines are automatically renumbered. Next, delete a range of lines from the following file:

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

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

```
,5D
```

The result is:

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

NAME Edit

PURPOSE Edits line of text.

SYNTAX [<line>]

REMARKS When a line number is typed, EDLIN displays the line number and text and reprints the line number on the line below. The line is now ready for editing. You may use any of the EDLIN editing commands to edit the line. The existing text of the line serves as the template until the **RETURN** key is pressed. If no line number is typed (i.e., if only **RETURN** 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 **RETURN** to accept the line as is.

Warning: If **RETURN** is pressed while the cursor is in the middle of the line, the remainder of the line is deleted.

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

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

To edit line 4, type:

```
4
```

The line is displayed with a cursor below it:

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

Now, using the **CTRL** 3 special editing key, type:

CTRL DEL number	4: number_
CTRL 3 **RETURN**	4: number four. 5:*_

NAME End

PURPOSE Ends the editing session.

SYNTAX E

REMARKS This command saves the edited file on disk, renames the original input file <filename>.BAK, and 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 a drive on which to save the file. The drive you want to save the file on must be selected when the editing session is started. If the drive is not selected when EDLIN is started, the file is saved on the disk in the default drive. The file may be copied to a different drive using the MS-DOS COPY command.

You also must be sure that the disk contains enough free space for the entire file. If the disk does not contain enough free space, the write is aborted and the edited file lost, although part of the file might be written out to the disk. For example:

E RETURN

After execution of the E command, the MS-DOS default drive prompt (e.g., A>) is displayed.

NAME Insert

PURPOSE Inserts text immediately before the specified <line>.

SYNTAX [<line>]I

REMARKS If you are creating a new file, the I command must be given before text can be typed (inserted). Text begins with line number 1. Successive line numbers appear automatically each time **RETURN** is pressed. EDLIN remains in insert mode until **CTRL C** is typed. When the insert is complete and insert mode has been exited, the line immediately following the inserted line(s) becomes the current line. All line numbers following the inserted section are incremented by the number of lines inserted.

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

For 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:*Use Delete and Insert
4: (the D and I commands)
5: to edit text
6: in your file.
```

To insert text before a specific line that is not the current line, type <line>I. For example:

```
6I
```

The result is:

```
6:_
```

Now, type the new text for line 6:

```
6: and renumber lines
```

To end the insertion, press **CTRL Z** and **RETURN** on the next line:

```
7: CTRL Z
```

Now type **L** to list the file. The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: Use Delete and Insert
4: (the D and I commands)
5: to edit text
6: and renumber lines
7:*in your file.
```

To insert lines immediately before the current line, type **I**. The result is:

```
7: _
```

Now, insert the following text and terminate with a **CTRL Z** on the next line:

```
7: so they are consecutive
8: CTRL Z
```

To list the file and see the result, type **L**. The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: Use Delete and Insert
4: (the D and I commands)
5: to edit text
6: and renumber lines
7: so they are consecutive
8:*in your file.
```

To append new lines to the end of the file, type **9I**. This produces the following:

```
9: _
```

Now, type the following new lines:

```
 9: The insert command can place new lines
10: in the file; there's no problem
11: because the line numbers are dynamic;
12: they'll go all the way to 65533.
```

End the insertion by pressing **CTRL Z RETURN** on line 13. The new lines appear at the end of all previous lines in the file. Now type the List command, **L**. The result is:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: Use Delete and Insert
 4: (the D and I commands)
 5: to edit text
 6: and renumber lines
 7: so they are consecutive
 8: in your file.
 9: The insert command can place new lines
10: in the file; there's no problem
11: because the line numbers are dynamic;
12: they'll go all the way to 65533.
```

NAME **List**

PURPOSE Lists a range of lines, including the two lines specified.

SYNTAX [<lstline>][,<endline>]L

REMARKS Default values are provided if either one or both of the options are omitted. If you omit the first option (e.g., ,**<endline>L**), the display starts 11 lines before the current line and ends with the specified <endline>. The beginning comma is required to indicate the omitted first option.

If you omit the second option, as in:

```
<lstline>L
```

23 lines are displayed, starting with the specified <lstline>. If you omit both parameters, as in:

```
L
```

23 lines are displayed: the 11 lines before the current line, the current line, and the 11 lines after the current line. If fewer than 11 lines are before the current line, more than 11 lines after the current line are displayed to make a total of 23 lines.

For 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: Use Delete and Insert
 4: (the D and I commands)
  ...
14: The current line remains unchanged.
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 <lstline>,<endline>L:

```
2,4L
```

The result is:

```
2: used to show dynamic line numbers.
3: Use Delete and Insert
4: (the D and I commands)
```

To list a range of 23 lines centered around the current line, type only L:

```
L
```

The result is:

```
 4: (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
```

NAME **Move**

PURPOSE Moves a range of text to lines immediately preceding <toline>.

SYNTAX <lstline>,<endline>,<toline>M

REMARKS Use the Move command to move a block of text (<lstline> to <endline>) to another location in the file. The lines are renumbered after the move.

For example, to move the text in lines 1 through 25 to before line 100, type:

```
1,25,100M
```

If the line numbers overlap, EDLIN displays an "Entry error" message. For another example, to move lines 80-100 to before line 30, type:

```
80,100,30M
```

NAME **Page**

PURPOSE Pages through a file 23 lines at a time.

SYNTAX [<lstline>][,<endline>]P

REMARKS If <lstline> is omitted, that number defaults to the current line plus one. If <endline> is omitted, 23 lines are listed.

The last line displayed becomes the new current line and is marked with an asterisk.

NAME Quit

PURPOSE Quits the editing session, does not save any editing changes, and exits to the MS-DOS operating system.

SYNTAX Q

REMARKS 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. See 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. For example:

```
Q
Abort edit (Y/N)?Y RETURN
A>_
```

NAME Replace

PURPOSE Replaces all occurrences of a string of text in the specified range with a different string of text or blanks.

SYNTAX [<lstline>][,<endline>][?]R<stringl>CTRL Z<string2>

REMARKS Each occurrence of <stringl> is replaced by <string2> and the line is displayed. If a line contains two or more replacements of <stringl> with <string2>, the line is displayed once for each occurrence. When all occurrences of <stringl> in the specified range are replaced by <string2>, the R command terminates and the asterisk prompt reappears.

<Stringl> must be separated from <string2> with a **CTRL Z**. <String2> may be ended with a **CTRL Z RETURN** combination or **RETURN**. If <stringl> is omitted, Replace takes the old (previously specified) <stringl> as its value.

If <lstline> is omitted in the range argument, <lstline> defaults to the line after the current line. If <endline> is omitted, it defaults to #, which indicates the line after the last line of the file.

If <string1> is ended with a **CTRL Z** and <string2> is a blank, <string1> is replaced with blanks. For example:

```
R<string1>CTRL Z RETURN
```

If the question mark (?) option is given, the Replace command stops at each line with a string that matches <string1> and displays the line with <string2> in place with the prompt **O.K.?**. If you press **Y** or **RETURN**, <string2> replaces <string1>, the next occurrence of <string1> is found, and the **O.K.?** prompt is displayed. After the last occurrence of <string1> is found in the range or file, EDLIN displays the asterisk prompt.

If you press any key besides **Y** or **RETURN** after the **O.K.?** prompt, <string1> is left unchanged and Replace goes to the next occurrence of <string1>. If <string1> occurs more than once in a line, each occurrence of <string1> is displayed individually with the **O.K.?** prompt. In this way, only the desired <string1> is replaced and you can prevent unwanted substitutions.

For example, 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: Use Delete and Insert
4: (the D and I commands)
5: to edit text
6: in your file.
7: The insert command can place new lines
8: in the file; there's no problem
9: because the line numbers are dynamic.
```

To replace all occurrences of <string1> with <string2> in a specified range without being prompted, type:

```
2,9 RandCTRL ZorRETURN
```

The result is:

```
3: Use Delete or Insert
4: (the D or I commands)
4: (the D or I commors)
7: The insert commor can place new lines
```

In the above replacement, some unwanted substitutions have occurred. To avoid these and to confirm each replacement, include a ? after the range. If only lstline is given with a ? option, you will be prompted for each replacement throughout the rest of the file. For example, to replace only certain occurrences of <string1> with <string2> using the original file, type:

```
2? RandCTRL ZorRETURN
```

The result is:

```
3: Use Delete or Insert
O.K.? Y
4: (The D or I commands)
O.K.? Y
4: (The D or I commors)
O.K.? N
7: The insert commor can place new lines
O.K.? N
*_
```

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

```
   ...
3: Use Delete or Insert
4: (The D or I commands)
   ...
7: The insert command can place new lines
   ...
```

Note that typing blanks on both sides of a string specifies whole words only. For example:

```
R and CTRL Z or RETURN
```

The word "and" is replaced by the word "or".

NAME Search

PURPOSE Searches the specified range of lines for a specified string of text.

SYNTAX [<lstline>][,<endline>][?]S<string>RETURN

REMARKS The <string> must be ended with a **RETURN**. The first line that matches <string> is displayed and becomes the current line. If the question mark option is not specified, the Search command terminates when a match is found. If no line contains a match for <string>, the message "Not found" is displayed.

If the question mark option (?) is included in the command, EDLIN displays the first line with a matching string and prompts you with the message **O.K.?**. If you press either **Y** or **RETURN**, the line becomes the current line and the search terminates. If you press any other key, the search continues until another match is found or until all lines have been searched and the "Not found" message is displayed.

If <lstline> is omitted it defaults to the line after the current line. If the <endline> is omitted it defaults to # (line after last line of file), which is the same as <lstline>,# S<string>. If <string> is omitted, Search takes the old (previously specified) string if there is one. If there is not an old string (i.e., no previous search or replace has been done), the command terminates immediately.

For example, 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: Use Delete and Insert
4: (the D and I commands)
5: to edit text
6: in your file.
7: The insert command can place new lines
8: in the file; there's no problem
9: because the line numbers are dynamic.
```

To search for the first occurrence of the string "and", type (with the **RETURN** key in parentheses):

```
2,9 Sand(RETURN)
```

The following line is displayed:

```
3: Use Delete and Insert
```

To search for another "and", modify the search command. Press **DEL** once to delete <lstline>, press **CTRL** 3 to copy the template's command line, and then press **RETURN**:

```
DEL CTRL-3,9 Sand(RETURN)
```

The search then continues from the line after the current line (line 3) because no first line was given. The result is:

```
4: (the D and I commands)
```

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

```
1, ? Sand
```

The result is:

```
3: Use Delete and Insert
O.K.?_
```

If you press any key (except **Y** or **RETURN**), the search continues. Type **N** here:

```
O.K.? N
```

Continue:

```
4: (the D and I commands)
O.K.?_
```

Now press Y to terminate the search:

```
O.K.? Y
*_
```

As another example, to search for string XYZ without the verification (O.K.?), type:

```
SXYZ
```

EDLIN reports a match and continues to search for the same string when you issue the S command:

```
S
```

EDLIN reports another match. Continue to type **S** until EDLIN reports the string is not found.

Note that <string> defaults to any string specified by a previous Replace or Search command.

NAME **Transfer**

PURPOSE Inserts (merges) the contents of <filename> into the file currently being edited at <line>. If <line> is omitted, the current line is used.

SYNTAX [<line>]T<filename>

REMARKS This command may be used to put the contents of a file into another file or into text you are typing. The transferred text is inserted at the line number specified by <line> and the lines are renumbered.

NAME **Write**

PURPOSE Writes a specified number of lines to disk from the lines that are being edited in memory. Lines are written to disk beginning with line number 1.

SYNTAX [<n>]W

REMARKS This command is used 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. You then can load additional unedited lines from disk into memory using the Append command.

Note: If you do not specify the number of lines, lines are written until memory is 3/4 full. No action is taken if available memory is already more than 3/4 full. All lines are renumbered, so that the first remaining line becomes line number 1.

Error Messages

When EDLIN finds an error, one of the following error messages is displayed:

Cannot edit .BAK file—rename file

Cause: You attempted to edit a file with a filename extension of .BAK; this extension is reserved for backup copies.

Cure: If you need to edit a .BAK file, RENAME the file with a different extension or COPY the .BAK file and give it a different filename extension.

No room in directory for file

Cause: When you attempted to create a new file, either the file directory was full or you specified an illegal disk drive or filename.

Cure: Check the command line that started EDLIN for illegal filename and disk drive entries. If this command line contains no illegal entries, run the CHKDSK program for the specified disk drive. If the status report shows that the disk directory is full, remove the disk. Insert and format a new disk.

Entry Error

Cause: The last command typed contained a syntax error.

Cure: Retype the command with the correct syntax and press **RETURN.**

Line too long

Cause: During a Replace command, the string given as the replacement caused the line to expand beyond the limit of 253 characters. EDLIN aborted the Replace command.

Cure: Divide the long line into two lines and try the Replace command twice.

Disk Full—file write not completed

Cause: You gave the End command, but the disk did not contain enough free space for the whole file. EDLIN aborted the E command and returned you to the operating system. Some of the file may have been written to the disk.

Cure: Only a portion (if any) of the file has been saved and your corrections have been lost. You should probably delete that portion of the file and restart the editing session. Always be sure that the disk has sufficient free space for the file to be written to disk before you begin your editing session.

Incorrect DOS version

Cause: You attempted to run EDLIN under a version of MS-DOS that was not 2.0 or higher.

Cure: The version of MS-DOS you use must be 2.0 or higher.

Invalid drive name or file

Cause: You have not specified a valid drive or filename when starting EDLIN.

Cure: Specify the correct drive or filename.

Filename must be specified

Cause: You did not specify a filename when you started EDLIN.

Cure: Specify a filename.

Invalid Parameter

Cause: You did not specify the /B switch when starting EDLIN.

Cure: Specify the /B switch when you start EDLIN.

Insufficient memory

Cause: There is not enough memory to run EDLIN.

Cure: You must free some memory by writing files to disk or by deleting files before restarting EDLIN.

File not found

Cause: The filename specified during a Transfer command was not found.

Cure: Specify a valid filename when issuing the Transfer command.

Must specify destination number

Cause: A destination line number was not specified for a Copy or Move command.

Cure: Reissue the command with a destination line number.

Not enough room to merge the entire file

Cause: There was not enough room in memory to hold the file during a Transfer command.

Cure: Free some memory by writing some files to disk or by deleting some files before you can transfer this file.

File creation error

Cause: The EDLIN temporary file cannot be created.

Cure: Ensure that the directory has enough space to create the temporary file and that the filename is not the name of a subdirectory in the edited file's directory.

DOS Guide

FC - File Comparison Utility

File Comparison Utility

Overview

You occasionally may wish to compare files on your disk. For example, you may want to compare two copies of a file to see which one is current. The MS-DOS File Comparison (FC) utility compares the contents of two files. The differences between the two files can be printed to the console or to a third file. The files may be source files (source statements of a programming language) or binary files (output from the MACRO-86 assembler or the MS-LINK Linker utility).

Comparisons may be made on a line-by-line or a byte-by-byte basis. The line-by-line comparison isolates and prints blocks of lines that differ between two files. The byte-by-byte comparison displays bytes that differ between files.

Limitations On Source Comparisons

FC uses a large amount of memory as buffer (storage) space to hold source files. If the source files are larger than available memory, FC compares the portions that can be loaded into the buffer space. If no lines match in the portions of the files in the buffer space, FC displays 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 recorded in the same manner as those files that fit completely in memory.

File Specifications

All file specifications use the following syntax:

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

where d: designates a disk drive (if the drive designation is omitted, FC uses the current default drive), filename is a one- to eight-character name of the file, and .ext is a one- to three-character extension to the filename.

How to Use FC

The syntax of FC is as follows:

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

FC matches the first file (filename1) against the second (filename2) and reports any differences between them. Both filenames can be pathnames, as shown in the following example:

```
FC B:\USER\MARY\FILE1.TXT \USER\FILE2.TXT
```

FC takes FILE1.TXT in the \USER\MARY directory of the disk in Drive B and compares it with FILE2.TXT in the \USER directory. Because no drive is specified for filename2, FC assumes that the \USER directory is on the disk in the default drive.

FC Switches

You can use four switches with the File Comparison utility: /B, /#, /W, and /C. They are described below:

/B Forces a binary comparison of both files. The two files are compared byte-to-byte; no attempt is made 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 (e.g., if file1 ends before file2), FC displays:

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

/# where # is a number from 1 to 9. This switch specifies the number of matching lines required for the files to be considered matched after a difference has been found. If this switch is not specified, the number of matching lines defaults to 3. This switch is used only in source comparisons.

/W Causes FC to compress tabs and spaces (whites) during the comparison. Thus, multiple contiguous whites in any line are considered a single white space. This switch is used only in source comparisons. Note that FC compresses whites, not ignores them. The exceptions are beginning and ending whites in a line, which are ignored. In the following example, an underscore represents a blank space:

```
___More__data_to_be_found____
```

matches with:

```
More_data_to_be_found
```

and with:

```
_______More_______data_to_be_____found_______
```

but does not match with:

```
____Moredata_to_be_found
```

/C Causes the matching process to ignore the case of letters so that all letters in the files are considered uppercase letters. This switch is used only in source comparisons. For example:

```
Much_MORE_data_IS_NOT_FOUND
```

matches:

```
much_more_data_is_not_found
```

If both the /W and /C options are specified, FC compresses blanks and ignores case. For example:

```
___DATA_was__found____
```

matches:

```
data_was_found
```

Difference Reporting

The File Comparison utility reports the differences between the two files you specify by displaying the first filename, the lines that differ between the files, and the first line to match in both files. FC then displays the name of the second file, the lines that are different, and the first line that matches in both files.

The default for the number of lines to match between files is 3 (to change this default, see the /# switch above).

The following example shows this format:

```
          ...

----------<filename1>
<difference>
<1st line to match file2 in file1>

----------<filename2>
<difference>
<1st line to match file1 in file2>

*****************************************

          ...
```

If more than /# lines are different, the program simply reports that the files are different and stops.

If no matches are found after the first difference, FC displays:

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

and returns to the MS-DOS default drive prompt (e.g., A>).

Redirecting FC Output to a File

The differences and matches between the specified files are displayed on your screen unless you redirect the output to a file. This is accomplished in the same way as MS-DOS command redirection, discussed in Chapter 4.

To compare File1 and File2 and send the FC output to a file called DIFFER.TXT, type:

```
FC File1 File2 >DIFFER.TXT
```

The differences and matches between File1 and File2 are written to DIFFER.TXT on the default drive.

Examples

Example 1

Assume these two ASCII files are on disk:

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

To compare the two files and display the differences on the terminal screen, type:

```
FC ALPHA.ASM BETA.ASM
```

FC compares ALPHA.ASM with BETA.ASM and displays the differences on the terminal screen. All other defaults (do not use tabs, spaces, or comments for matches, and do a source comparison on the two files) remain intact.

The output appears as follows on the terminal screen (the Notes do not appear):

```
----------ALPHA.ASM
D                       Note: ALPHA file
E                       contains defg,
F                       BETA contains g.
G

----------BETA.ASM
G

*****************************

----------ALPHA.ASM
M                       Note: ALPHA file
N                       contains mno where
O                       BETA contains jl2.
P

----------BETA.ASM
J
1
2
P

*****************************

----------ALPHA.ASM
W                       Note: ALPHA file
                        contains w where
----------BETA.ASM       BETA contains 45w.
4
5
W
```

Example 2

You can print the differences on a printer using the same two source files. In this example, four successive lines must be the same to constitute a match:

```
FC /4 ALPHA.ASM BETA.ASM >PRN
```

The following output should appear on the printer:

```
----------ALPHA.ASM
D
E
F
G
H
I
M
N                    Note: p is the lst of
O                    a string of 4 matches.
P
----------BETA.ASM
G
H
I
J
1
2
P

--------------------------------

----------ALPHA.ASM
W
                     Note: w is the lst of a
----------BETA.ASM    string of 4 matches.
4
5
W
```

Example 3

This example forces a binary comparison through the /B switch and displays the differences on the terminal screen using the same two source files as were used in the previous examples:

```
FC /B ALPHA.ASM BETA.ASM
```

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

*** Data left in F1 ***
```

Error Messages

When the File Comparison utility detects an error, one or more of the following error messages are displayed:

Incorrect DOS version

Cause: You are running FC under a version of MS-DOS that is not 2.0 or higher.

Invalid parameter:<option>

Cause: One of the switches you specified is invalid.

File not found:<filename>

Cause: FC could not find the filename you specified.

Read error in:<filename>

Cause: FC could not read the entire file.

Invalid number of parameters

Cause: You have specified the wrong number of options on the FC command line.

DOS Guide

Configuring Your System and Adding Drivers

Configuring Your System and Adding Drivers

Overview

When MS-DOS boots, it checks the root directory for a configuration file named "CONFIG.SYS" and executes the commands within this file. This file allows you to add device drivers and to customize your system by changing preset DOS software settings. Additions or changes to CONFIG.SYS will be in effect the next time DOS is cold booted.

The following list of commands can be present in CONFIG.SYS:

- AVAILDEV specifies how DOS references a device,
- BREAK=ON/OFF specifies DOS's treatment of break codes,
- BUFFERS=xx specifies the number of disk buffers,
- DEVICE specifies the filename of a device driver,
- FILES=xx specifies the number of files opened via disk handles,
- SHELL specifies a command processor,
- SWITCHAR specifies the device switch character.

Creating the CONFIG.SYS File

The configuration file must be created by you (through a word processor such as EDLIN), assigned the name CONFIG.SYS, and saved in the MS-DOS disk root directory. This file can contain any of the commands described in the following sections.

Available Device

AVAILDEV = <TRUE or FALSE> (default is TRUE)

Available device specifies how DOS accepts device driver names. AVAILDEV TRUE allows DOS to accept both /dev/<dev> and <dev> as valid device references. If FALSE is selected, only /dev/<dev> references the device, and <dev> references a filename.

Assign Break Code Performance

BREAK=ON/OFF (default is OFF)

BREAK=OFF causes DOS to check for **CTRL C** entered at the keyboard only when DOS is accessing the screen, the keyboard, or a peripheral device. Therefore, you may not be able to cancel an executing program with **CTRL C** unless the program uses DOS to access an I/O device.

BREAK=ON checks for **CTRL C** whenever DOS performs any function for a program. This allows you to break out of any program, including programs that perform few I/O accesses, such as compilers.

Allocate Disk Buffers

BUFFERS=xx (where xx is 1 - 99; default is 2)

The BUFFERS command specifies the number of disk buffers to allocate in memory at start up.

The more disk buffers within DOS, the more data can be stored in memory, resulting in faster disk response time. When DOS is requested to transfer a record, DOS checks to see if the record is already in a buffer. If the record is in a buffer, DOS simply transfers the record to the application's area without having to do a disk access.

This greatly enhances disk transfer's speed performance when using random files, such as during most data base applications. However, sequential read and write applications (read an entire file, write an entire file) are not enhanced by a greater number of buffers.

For most data base applications, a value between 10 and 20 buffers generally provides the best results. When more than 20 buffers are allocated, DOS can take longer to search all buffers than to just read the record from disk. Also note that the size of memory for user applications decreases by 528 bytes per buffer.

A minimum of three disk buffers is recommended for Attaches with the hard disk option.

Assign New Peripheral Devices

DEVICE=[d:][path]<filename>[.ext]

where [d:] is the drive reference,
[path] is the directory or subdirectory,
<filename> is the name of the device driver file,
[.ext] is the file type.

Parameters enclosed in brackets ([ ]) are optional.

The DEVICE command specifies the name of a file which contains a device driver. DOS loads this file into memory as a DOS extension and allows the drive to be utilized. (Drivers for the screen, keyboard, printer, auxiliary device, diskette, and hard disk devices are automatically loaded.)

User-written device drivers should include one DEVICE= command for each driver to be loaded at system start up. Information about writing device drivers is contained in subsequent sections in this chapter.

Assigning File Opens via Handles

FILES=xx (where xx is 1 - 99; default is eight)

The FILES command specifies the maximum number of open files that the XENIX system calls can access using DOS handles. The number of files opened through standard File Control Blocks (FCBs) is not affected by FILES.

Application programs can specify an ASCII string consisting of the drive reference, pathname, and filename to open or create a file instead of constructing an FCB. DOS locates this drive, directory, and file and returns a 16-bit binary value called a "handle". All subsequent file accesses can be performed by identifying the handle to DOS.

The default value of eight files is usually sufficient for most operations. A FILES statement to increase this value is only required if an application results in an error message indicating an insufficient number of handles.

If you specify more than eight files, the resident portion of DOS increases by 39 bytes per file, reducing user-available memory by the same amount.

Assigning a Different Command Processor

```
SHELL=[d:][path]<filename>[.ext]
```

where [d:] is the drive reference,
[path] is the directory or subdirectory,
<filename> is the name of the command processor file,
[.ext] is the file type.

The SHELL command allows you to specify the name and location of a top-level command processor to load instead of COMMAND.COM. This command processor should handle interrupts 22 {hex}, 23 {hex}, 24 {hex}, and read and execute internal commands. The batch processor and program loader reside in COMMAND.COM, so their functions must be emulated by your processor.

Assign Switch Characters

```
SWITCHAR=<char> (default is /)
```

SWITCHAR causes DOS to use <char> as the current switch designator.

CONFIG.SYS Example

A typical configuration file might look like this:

```
Buffers=10
Files=10
Device=\BIN\NETWORK.SYS
Break=ON
Switchar=-
Shell=A:\BIN\COMMAND.COM A:\BIN /P
```

The Buffers and Files parameters in this example are set to 10. The system initialization routine searches for the filename \BIN\NETWORK.SYS and adds its defined device to the system. The Device statement must specify the correct path for the driver.

This configuration file also sets the MS-DOS command EXEC to the COMMAND.COM file located in the \BIN directory on the disk in Drive A. A:\BIN tells DOS where to look for COMMAND.COM when it must be read from disk. /P indicates that COMMAND.COM is the first program running on the system to allow COMMAND.COM to process the MS-DOS EXIT command.

Device Drivers

A device driver is a binary file containing code that manipulates its device's hardware and interfaces with MS-DOS. In addition, it contains a header that identifies the file as a device driver, defines the strategy and interrupt entry points, and describes the various device attributes.

Note: The file must not use "ORG 100H" like .COM files. Because it does not use the Program Segment Prefix, the device driver is simply loaded and must have an origin of zero (ORG 0 or no ORG statement).

There are two kinds of device drivers: character device drivers and block device drivers. Character devices perform serial character I/O, such as CON, AUX, and PRN. These devices are given specific names (such as CON or AUX); users may open I/O channels using either handles or FCBs.

Block devices are the "disk drives" on the system. They can perform random I/O in pieces called blocks (usually the physical sector size). These devices are not named as the character devices are, and therefore cannot be opened directly. Instead, they are identified through the drive letters (A, B, C, etc.).

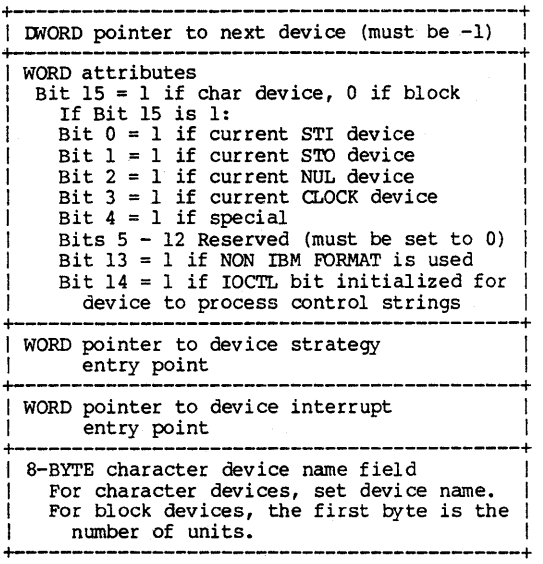
Block devices use "units" which represent the disk drives. A single driver may be responsible for one or more disk drives. For example, block device driver ALPHA may be responsible for drives A, B, C, and D. This means that it has four units (0-3) defined and uses four drive letters.

The position of the driver in the list of all drivers determines which units correspond to which disk drive letters. If driver ALPHA is the first block driver in the device list with units (0-3), disk drive letters are A, B, C, and D. If BETA is the second block driver which defines two units (0-1), drive letters are E and F, and so on. DOS is limited to 63 block device units, although after 26 characters the drive letters are unconventional, such as] and /.

Note that character devices cannot define multiple units because they have only one name.

Device Headers

A device header is required at the beginning of each device driver. Device headers use the following format:



A WORD is two bytes. DWORD is a double word (offset followed by segment). Device entry points are words. They must be offsets from the same segment number that points to this table. For example, if XXX:YYY points to the start of this table, XXX:strategy and XXX:interrupt are the entry points.

Pointer to Next Device Field

The pointer to the next device header field is a double word field (offset followed by segment) set by MS-DOS when the driver is loaded to point at the next driver in the system list. This field must be set to -1 prior to being loaded (when it is on the disk as a file) unless more than one device driver is in the file. If more than one driver is in the file, the first word of the double word pointer should be the offset of the next driver's Device Header.

If there is more than one device driver in the .COM file, the last driver in the file must have the pointer to the next Device Header field set to -1.

Attribute Field

The attribute field (Bit 15) identifies to the system whether this device is a block or character device. Additional bits in the attribute field assign special treatment for selected character devices (these bits are ignored for a block device).

For example, you might have a new device driver that you want as the standard input and output device. In addition to installing the driver, you must indicate to DOS that you want this new driver to override the current standard input and standard output (the CON device). This is accomplished by setting the attributes to the desired characteristics, so you would set Bit 0 (STI) and Bit 1 (STO) to 1. Similarly, a new CLOCK device could be installed by setting that attribute (Bit 3 = 1).

Although there is a NUL device attribute (Bit 2), the NUL device cannot be reassigned. This attribute exists so that MS-DOS can determine if the NUL device is being used.

The NON IBM FORMAT bit (Bit 13) applies only to block devices and affects the operation of the BUILD BPB (BIOS Parameter Block) device call.

IOCTL (Input/Output Control) functions allow the device to transfer data for its own use (for example, to set baud rate, stop bits, and form length) instead of passing data over the device channel, as during a normal read or write. Though interpretation of the passed information is up to the device, the data must not be treated as a normal I/O request.

The IOCTL bit (Bit 14) applies to both character and block devices. This bit tells MS-DOS whether the device can handle control strings via the IOCTL system call, Function 44H.

If a driver cannot process control strings, it should initially set Bit 14 to 0. This tells MS-DOS to return an error if an attempt is made (via Function 44H) to send or receive control strings to this device.

A device which can process control strings should initialize the IOCTL bit to 1. For these drivers, MS-DOS makes calls to the IOCTL INPUT and OUTPUT device functions to send and receive IOCTL strings.

Strategy and Interrupt Routines

These two fields are the pointers to the entry points of the strategy and interrupt routines. They are word values, so they must be in the same segment as the Device Header.

Name Field

This is an 8-byte field that contains the name of a character device or the number of units of a block device. For a block device, the number of units can be put in the first byte. This is optional, because MS-DOS fills in this location with the value returned by the driver's INIT code.

Create a Device Driver

In order to create a device driver that MS-DOS can install, you must write a binary file with a Device Header at the beginning of the file. For device drivers, the code must be originated at zero, not 100H. The link field (pointer to the next Device Header) should be -1, unless there is more than one device driver in the file. The attribute field and entry points must be set correctly.

If the driver is for a character device, the name field should be filled in with the name of that character device. The name can be any legal 8-character filename.

MS-DOS always processes installable device drivers before handling the default devices; to install a new CON device, simply name the device CON. Set the standard input device and standard output device bits in the attribute word on a new CON device. The scan of the device list stops on the first match, so the installable device driver takes precedence.

MS-DOS can install the driver anywhere in memory, so use care when making any far memory references. You should not expect your driver to always be loaded in the same place every time.

Installing Device Drivers

MS-DOS allows new device drivers to be installed dynamically at boot time. This is accomplished by INIT code in the BIOS, which reads and processes the CONFIG.SYS file.

MS-DOS calls upon the device drivers to perform their function by making a far call to strategy entry and passing (in a Request Header) information which describes the functions of the device driver.

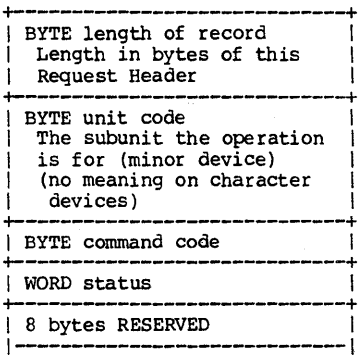
This structure allows you to program an interrupt-driven device driver. For example, you may want to perform local buffering in a printer.

Request Header

When MS-DOS calls a device driver to perform a function, it passes a Request Header in ES:BX to the strategy entry point. This is a fixed length header, followed by data pertinent to the operation being performed.

The device driver is responsible for preserving the machine state, such as saving all registers on entry and restoring them on exit. When strategy or interrupt is called, the stack provides room for approximately 20 pushes. If more stack room is needed, the driver should set up its own stack.

The following example illustrates a Request Header:



Unit Code

The unit code field identifies which unit in your device driver the request is for. For example, if your device driver has 3 units defined, the possible values of the unit code field would be 0, 1, and 2.

Command Code Field

The command code field in the Request header can have the following values:

Command Code	Function
0	INIT
1	MEDIA CHECK (Block only, NOP for character)
2	BUILD BPB " " " " "
3	IOCTL INPUT (Only called if device has IOCTL)
4	INPUT (read)
5	NON-DESTRUCTIVE INPUT NO WAIT (Char devs only)
6	INPUT STATUS " " "
7	INPUT FLUSH " " "
8	OUTPUT (write)
9	OUTPUT (Write) with verify
10	OUTPUT STATUS " " "
11	OUTPUT FLUSH " " "
12	IOCTL OUTPUT (Only called if device has IOCTL)

MEDIA CHECK and BUILD BPB (BIOS Parameter Block)

MEDIA CHECK and BUILD BPB are used with block devices only. MS-DOS first calls MEDIA CHECK for diskette accesses and passes its current media descriptor byte (as described in a later section).

MEDIA CHECK returns one of the following results:

Media Not Changed Current parameter block and media byte are OK.

Media Changed Current parameter block and media are wrong. MS-DOS invalidates any buffers for this unit and calls the device driver to build the BPB with media byte and buffer.

Not Sure If there are dirty buffers (buffers with changed data not yet written to disk) for this unit, MS-DOS assumes the parameter block and media byte are OK (media not changed). If nothing is dirty, MS-DOS assumes the media has changed. It invalidates any buffers for the unit, and calls the device driver to build the BPB with media byte and buffer.

Error DOS sets an error code to identify the error.

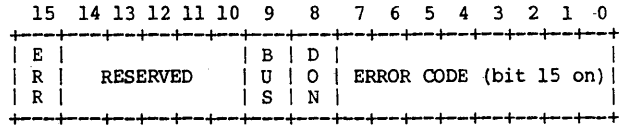
MS-DOS calls BUILD BPB if "Media Changed" is returned, or if "Not Sure" is returned and there are no dirty buffers.

The BUILD BPB call also gets a pointer to a one-sector buffer. This buffer's contents are determined by the NON IBM FORMAT bit in the attribute field. If the bit is zero (device is IBM format-compatible), the buffer contains the first sector of the first File Allocation Table (FAT). The FAT ID byte is the first byte of this buffer.

Note: The BPB must locate the FAT at the same address for all possible media because this first FAT sector must be read before the actual BPB is returned. If the NON IBM FORMAT bit is set, the pointer points to one sector of scratch space (which may be used for anything).

Status Word

The following example illustrates the status word in the Request Header.



The status word is zero on entry and is set by the driver interrupt routine on return.

Bit 8 is the done bit. When set, it means the operation is complete. The driver sets it to 1 when it exits.

Bit 15 is the error bit. If it is set, the low 8 bits indicate the error. The errors are:

0 Write protect violation
1 Unknown Unit
2 Drive not ready
3 Unknown command
4 CRC error
5 Bad drive request structure length
6 Seek error
7 Unknown media
8 Sector not found
9 Printer out of paper
A Write fault
B Read Fault
C General failure

Bit 9 is the busy bit, which is set only by status calls.

For output on character devices: If bit 9 is 1 on return, a write request waits for completion of a current request. If bit 9 is 0, there is no current request and a write request starts immediately.

For input on character devices with a buffer: If bit 9 is 1 on return, a read request goes to the physical device. If bit 9 is 0 on return, there are characters in the device buffer and a read returns quickly.

Bit 9 at zero also indicates that something has been typed. MS-DOS assumes all character devices have an input type-ahead buffer. Devices that do not have a type-ahead buffer should always return busy=0 so MS-DOS does not continuously wait for input to enter a nonexistent buffer.

INIT is a function defined for each device. This routine is called only once, when the device is installed. The INIT routine returns a location (DS:DX) which is a pointer to the first free byte of memory after the device driver. This pointer method can be used to delete initialization code that is only needed once, which saves space.

Block devices are installed the same way as character devices, and also return a first free byte pointer as described above. Additional information is returned as listed below:

Number of units

This determines logical device names. For example, if the current maximum logical device letter is F at the time of the install call and the INIT routine returns 4 as the number of units, devices have logical names G, H, I, and J. This mapping is determined by the position of the driver in the device list and by the number of units on the device (stored in the first byte of the device name field).

Pointer to a BPB (BIOS Parameter Block) pointer array

There is one table for each unit defined. These blocks are used to build an internal DOS data structure for each unit. The pointer passed to the DOS from the driver points to an array of n word pointers to BPBs, where n is the number of units defined. If all units are the same, all of the pointers can point to the same BPB. This array must be protected (below the free pointer set by the return) because an internal DOS structure is built starting at the byte pointed to by the free pointer. The sector size defined must be less than or equal to the maximum sector size defined at default BIOS INIT time, or the install fails.

Media descriptor byte

INIT of a block device must pass back this byte last. This byte is passed to devices so that they know what parameters MS-DOS is currently using for a particular drive unit.

Block devices may be "dumb" or "smart". A dumb device defines a unit (and therefore an internal DOS structure) for each possible media drive combination. For example, unit 0 = drive 0 single side, unit 1 = drive 0 double side. For this approach, media descriptor bytes do not mean anything.

A smart device allows multiple media per unit. In this case, the BPB table returned at INIT must define space large enough to accommodate the largest possible media supported. Smart drivers use the media descriptor byte to pass information about what media is currently in a unit.

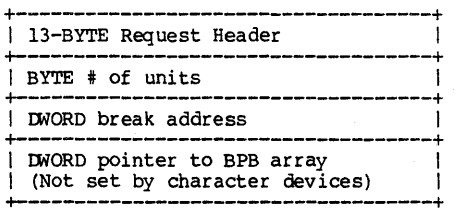
Function Call Parameters

All strategy routines are called with ES:BX pointing to the Request Header. The interrupt routines get the pointers to the Request Header from the queue in which the strategy routines store them. The command code in the Request Header tells the driver which function to perform.

Note: All DWORD pointers are stored offset first, then segment.

INIT **Command code = 0**

INIT - ES:BX ->

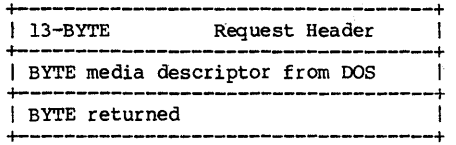


The number of units, break address, and BPB pointer are set by the driver. On entry, the DWORD that is to be set to the BPB array (on block devices) points to the character after the '=' on the line in CONFIG.SYS that loaded this device. This allows drivers to scan the CONFIG.SYS invocation line for arguments.

Note: If multiple device drivers are in a single .COM file, the ending address returned by the last INIT called is the one MS-DOS uses. All of the device drivers in a single .COM file should return the same ending address.

MEDIA CHECK **Command Code = 1**

MEDIA CHECK - ES:BX ->



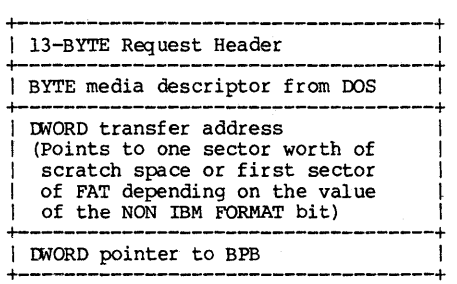
In addition to setting the status word, the driver must set the return byte to one of the following:

```
-1  Media has been changed
 0  Don't know if media has been changed
 1  Media has not been changed
```

If the driver can return -1 or 1 (by having a door-lock or other interlock mechanism), MS-DOS performance is enhanced because DOS does not need to reread the FAT for each directory access.

BUILD BPB (BIOS Parameter Block) Command code = 2

BUILD BPB - ES:BX ->



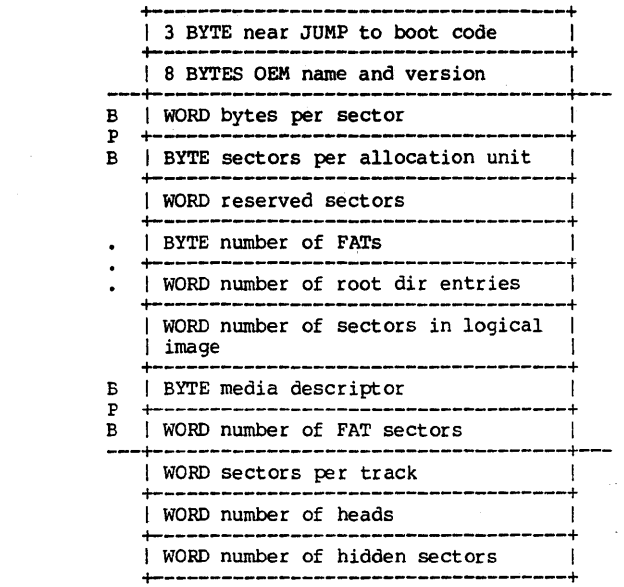
If the NON IBM FORMAT bit of the device is set, the DWORD transfer address points to a one-sector buffer which can be used for any purpose. If the NON IBM FORMAT bit is 0, this buffer contains the first sector of the first FAT and the driver must not alter this buffer.

If IBM-compatible format is used (NON IBM FORMAT BIT = 0), the first sector of the first FAT must be located at the same sector on all possible media. This is because the FAT sector is read BEFORE the media is actually determined. Use this mode if all you want is to read the FAT ID byte.

In addition to setting status word, the driver must set the Pointer to the BPB on return.

To allow different OEMs to read each other's disks, information relating to the BPB for a particular piece of media should be kept in the boot sector for the media.

The format of the boot sector is:



The last three words (sectors per track, number of heads, and number of hidden sectors) are optional, intended to help the BIOS understand the media. Sectors per track may be redundant (could be calculated from total size of the disk). Number of heads is useful for supporting different multi-head drives which have the same storage capacity but a different number of surfaces. The number of hidden sectors may be used to support drive-partitioning schemes.

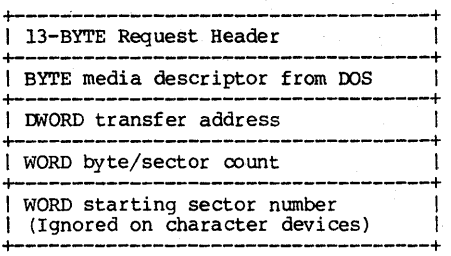
Media Descriptor Byte

The last two digits of the FAT ID byte are called the media descriptor byte. Currently, the media descriptor byte has been defined for a few media types, including 5-1/4" and 8" standard disks.

Although these media bytes map directly to FAT ID bytes (which are constrained to the eight values F8-FF), media bytes can be any value in the range 0-FF.

READ or WRITE Command codes = 3, 4, 8, 9, and 12

READ or WRITE - ES:BX (including IOCTL) ->



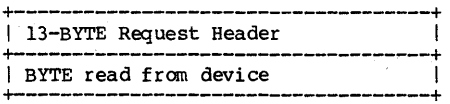
In addition to setting the status word, the driver must set the sector count to the actual number of sectors (or bytes) transferred. No error checking is performed on an IOCTL I/O call. The driver must correctly set the return sector (byte) count to the actual number of bytes transferred.

When using block devices, BIOS may be asked to perform a write operation of 64K bytes which appear to be a "wrap around" of the transfer address in the BIOS I/O packet. This request arises from an optimization added to the write code in MS-DOS. It only arises during user writes within a sector size of 64K bytes on files "growing" past the current EOF. BIOS can ignore the balance of the write that "wraps around" if it so chooses.

For example, a write of 10000H bytes worth of sectors with a transfer address of XXX:1 could ignore the last two bytes. A user program can never request an I/O of more than FFFFH bytes and cannot wrap around (even to 0) in the transfer segment. Therefore, the last two bytes can be ignored in this case.

NON DESTRUCTIVE READ NO WAIT Command code = 5

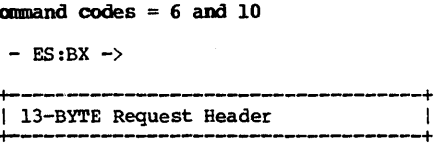
NON DESTRUCTIVE READ NO WAIT - ES:BX ->



If the character device returns busy bit = 0 (characters in buffer), the next character to be read is returned. This character is not removed from the input buffer (hence the term "Non Destructive Read"). Basically, this call allows MS-DOS to look ahead one input character.

STATUS Command codes = 6 and 10

STATUS Calls - ES:BX ->



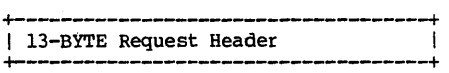
The driver must set the status word and the busy bit as follows:

For output on character devices: If bit 9 is 1 on return, a write request waits for completion of a current request. If bit 9 is 0, there is no current request and a write request starts immediately.

For input on character devices with a buffer: A return of 1 means a read request goes to the physical device. If bit 9 is 0 on return, characters are in the device's buffer and a read returns quickly. Zero also indicates that the user has typed something. MS-DOS assumes that all character devices have an input type-ahead buffer. Devices that do not have a type-ahead buffer should always return busy = 0 so that the DOS does not wait continuously for input to a nonexistent buffer.

FLUSH Command codes = 7 and 11

FLUSH Calls - ES:BX ->



The FLUSH call tells the driver to flush (terminate) all pending requests. This call is used to flush the input queue on character devices.

Sample Block Device Driver

```
;********************* A BLOCK DEVICE ******************

        TITLE   5 1/4" DISK DRIVER FOR SCP DISK-MASTER

;This driver is intended to drive up to four 5 1/4" drives
;hooked to the Seattle Computer Products DISK MASTER disk
;controller. All standard IBM PC formats are supported.


FALSE   EQU     0
TRUE    EQU     NOT FALSE

;The I/O port address of the DISK MASTER
DISK    EQU     0E0H
;DISK+0
;       1793    Command/Status
;DISK+1
;       1793    Track
;DISK+2
;       1793    Sector
;DISK+3
;       1793    Data
;DISK+4
;       Aux Command/Status
;DISK+5
;       Wait Sync

;Back side select bit
BACKBIT EQU     04H
;5 1/4" select bit
SMALBIT EQU     10H
;Double Density bit
DDBIT   EQU     08H

;Done bit in status register
DONEBIT EQU     01H

;Use table below to select head step speed.
;Step times for 5" drives
;are double that shown in the table.
;
;Step value     1771    1793
;
;    0           6ms     3ms
;    1           6ms     6ms
;    2          10ms    10ms
;    3          20ms    15ms
;
STPSPD  EQU     1
NUMERR  EQU     ERROUT-ERRIN
CR      EQU     0DH
LF      EQU     0AH
```

```
CODE    SEGMENT
ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
;--------------------------------------------------------
;
;       DEVICE HEADER
;
DRVDEV  LABEL   WORD
        DW      -1,-1
        DW      0000      ;IBM format-compatible, Block
        DW      STRATEGY
        DW      DRV$IN
DRVMAX  DB      4

DRVTBL  LABEL   WORD
        DW      DRV$INIT
        DW      MEDIA$CHK
        DW      GET$BPB
        DW      CMDERR
        DW      DRV$READ
        DW      EXIT
        DW      EXIT
        DW      EXIT
        DW      DRV$WRIT
        DW      DRV$WRIT
        DW      EXIT
        DW      EXIT
        DW      EXIT

;--------------------------------------
;
;       STRATEGY

PTRSAV  DD      0

STRATP  PROC    FAR
STRATEGY:
        MOV     WORD PTR [PTRSAV],BX
        MOV     WORD PTR [PTRSAV+2],ES
        RET
STRATP  ENDP


;---------------------------------------
;
;       MAIN ENTRY

CMDLEN  =       0       ;LENGTH OF THIS COMMAND
UNIT    =       1       ;SUB UNIT SPECIFIER
CMDC    =       2       ;COMMAND CODE
STATUS  =       3       ;STATUS
MEDIA   =       13      ;MEDIA DESCRIPTOR
TRANS   =       14      ;TRANSFER ADDRESS
COUNT   =       18      ;COUNT OF BLOCKS OR CHARACTERS
START   =       20      ;FIRST BLOCK TO TRANSFER
```

```
DRV$IN:
        PUSH    SI
        PUSH    AX
        PUSH    CX
        PUSH    DX
        PUSH    DI
        PUSH    BP
        PUSH    DS
        PUSH    ES
        PUSH    BX

        LDS     BX,[PTRSAV]     ;GET POINTER TO I/O PACKET

        MOV     AL,BYTE PTR [BX].UNIT   ;AL = UNIT CODE
        MOV     AH,BYTE PTR [BX].MEDIA  ;AH = MEDIA DESCRIP
        MOV     CX,WORD PTR [BX].COUNT  ;CX = COUNT
        MOV     DX,WORD PTR [BX].START  ;DX = START SECTOR
        PUSH    AX
        MOV     AL,BYTE PTR [BX].CMDC   ;Command code
        CMP     AL,11
        JA      CMDERRP                 ;Bad command
        CBW
        SHL     AX,1                    ;2 times command =
                                        ;word table index
        MOV     SI,OFFSET DRVTBL
        ADD     SI,AX                   ;Index into table
        POP     AX                      ;Get back media
                                        ;and unit

        LES     DI,DWORD PTR [BX].TRANS ;ES:DI = TRANSFER
                                        ;ADDRESS
        PUSH    CS
        POP     DS

ASSUME  DS:CODE

        JMP     WORD PTR [SI]             ;GO DO COMMAND

;--------------------------------------------------------
;
;       EXIT - ALL ROUTINES RETURN THROUGH THIS PATH
;
ASSUME  DS:NOTHING
CMDERRP:

        POP     AX                      ;Clean stack
CMDERR:
        MOV     AL,3                    ;UNKNOWN COMMAND ERROR
        JMP     SHORT ERR$EXIT

ERR$CNT:LDS     BX,[PTRSAV]
        SUB     WORD PTR [BX].COUNT,CX ;# OF SUCCESS. I/Os
```

```
ERR$EXIT:
;AL has error code
        MOV     AH,10000001B            ;MARK ERROR RETURN
        JMP     SHORT ERR1

EXITP   PROC    FAR

EXIT:   MOV     AH,00000001B
ERR1:   LDS     BX,[PTRSAV]
        MOV     WORD PTR [BX].STATUS,AX
                                        ;MARK OPERATION COMPLETE
        POP     BX
        POP     ES
        POP     DS
        POP     BP
        POP     DI
        POP     DX
        POP     CX
        POP     AX
        POP     SI
        RET                             ;RESTORE REGS AND RETURN
EXITP   ENDP

CURDRV  DB      -1

TRKTAB  DB      -1,-1,-1,-1

SECCNT  DW      0

DRVLIM  =       8       ;Number of sectors on device
SECLIM  =       13      ;MAXIMUM SECTOR
HDLIM   =       15      ;MAXIMUM HEAD

;WARNING - preserve order of drive and curhd!

DRIVE   DB      0       ;PHYSICAL DRIVE CODE
CURHD   DB      0       ;CURRENT HEAD
CURSEC  DB      0       ;CURRENT SECTOR
CURTRK  DW      0       ;CURRENT TRACK

;
MEDIA$CHK:              ;Always indicates Don't know
ASSUME  DS:CODE
        TEST    AH,00000100B        ;TEST IF MEDIA REMOVABLE
        JZ      MEDIA$EXT
        XOR     DI,DI               ;SAY I DON'T KNOW
MEDIA$EXT:
        LDS     BX,[PTRSAV]
        MOV     WORD PTR [BX].TRANS,DI
        JMP     EXIT

BUILD$BPB:
ASSUME  DS:CODE
        MOV     AH,BYTE PTR ES:[DI]     ;GET FAT ID BYTE
        CALL    GETBP                   ;TRANSLATE
```

```
SETBPB: LDS     BX,[PTRSAV]
        MOV     [BX].MEDIA,AH
        MOV     [BX].COUNT,DI
        MOV     [BX].COUNT+2,CS
        JMP     EXIT

BUILDBP:
ASSUME  DS:NOTHING
;AH is media byte on entry
;DI points to correct BPB on return
        PUSH    AX
        PUSH    CX
        PUSH    DX
        PUSH    BX
        MOV     CL,AH           ;SAVE MEDIA
        AND     CL,0F8H         ;NORMALIZE
        CMP     CL,0F8H         ;COMPARE WITH GOOD MEDIA BYTE
        JZ      GOODID
        MOV     AH,0FEH         ;DEFAULT TO 8-SECTOR,
                                ;SINGLE-SIDED
GOODID:
        MOV     AL,1            ;SET NUMBER OF FAT SECTORS
        MOV     BX,64*256+8     ;SET DIR ENTRIES AND SECTOR MAX
        MOV     CX,40*8         ;SET SIZE OF DRIVE
        MOV     DX,01*256+1     ;SET HEAD LIMIT & SEC/ALL UNIT
        MOV     DI,OFFSET DRVBPB
        TEST    AH,00000010B    ;TEST FOR 8 OR 9 SECTOR
        JNZ     HAS8            ;NZ = HAS 8 SECTORS
        INC     AL              ;INC NUMBER OF FAT SECTORS
        INC     BL              ;INC SECTOR MAX
        ADD     CX,40           ;INCREASE SIZE
HAS8:   TEST    AH,00000001B    ;TEST FOR 1 OR 2 HEADS
        JZ      HAS1            ;Z = 1 HEAD
        ADD     CX,CX           ;DOUBLE SIZE OF DISK
        MOV     BH,112          ;INCREASE # OF DIREC. ENTRIES
        INC     DH              ;INC SEC/ALL UNIT
        INC     DL              ;INC HEAD LIMIT
HAS1:   MOV     BYTE PTR [DI].2,DH
        MOV     BYTE PTR [DI].6,BH
        MOV     WORD PTR [DI].8,CX
        MOV     BYTE PTR [DI].10,AH
        MOV     BYTE PTR [DI].11,AL
        MOV     BYTE PTR [DI].13,BL
        MOV     BYTE PTR [DI].15,DL
        POP     BX
        POP     DX
        POP     CX
        POP     AX
        RET
```

```
;--------------------------------------------------------
;       DISK I/O HANDLERS
;
;ENTRY:
;       AL = DRIVE NUMBER (0-3)
;       AH = MEDIA DESCRIPTOR
;       CX = SECTOR COUNT
;       DX = FIRST SECTOR
;       DS = CS
;       ES:DI = TRANSFER ADDRESS
;EXIT:
;       IF SUCCESSFUL CARRY FLAG = 0
;         ELSE CF=1 AND AL CONTAINS (MS-DOS) ERROR CODE,
;         CX # sectors NOT transferred

DRV$READ:
ASSUME  DS:CODE
        JCXZ    DSKOK
        CALL    SETUP
        JC      DSK$IO
        CALL    DISKRD
        JMP     SHORT DSK$IO

DRV$WRIT:
ASSUME  DS:CODE
        JCXZ    DSKOK
        CALL    SETUP
        JC      DSK$IO
        CALL    DISKWRT
ASSUME  DS:NOTHING
DSK$IO: JNC     DSKOK
        JMP     ERR$CNT
DSKOK:  JMP     EXIT

SETUP:
ASSUME  DS:CODE
;Input same as above
;On output
; ES:DI = Trans addr
; DS:BX Points to BPB
; Carry set if error (AL is error code (MS-DOS))
; else
;       [DRIVE] = Drive number (0-3)
;       [SECCNT] = Sectors to transfer
;       [CURSEC] = Sector number of start of I/O
;       [CURHD]  = Head number of start of I/O   ;Set
;       [CURTRK] = Track # of start of I/O ;Seek performed
; All other registers destroyed

        XCHG    BX,DI                   ;ES:BX = TRANSFER ADDRESS
        CALL    GETBP                   ;DS:DI = PTR TO B.P.B
        MOV     SI,CX
        ADD     SI,DX
        CMP     SI,WORD PTR [DI].DRVLIM
                                        ;COMPARE AGAINST DRIVE MAX
```

```
        JBE     INRANGE
        MOV     AL,8
        STC
        RET

INRANGE:
        MOV     [DRIVE],AL
        MOV     [SECCNT],CX     ;SAVE SECTOR COUNT
        XCHG    AX,DX           ;SET UP LOGICAL SECTOR
                                ;FOR DIVIDE
        XOR     DX,DX
        DIV     WORD PTR [DI].SECLIM ;DIVIDE BY SEC PER TRACK
        INC     DL
        MOV     [CURSEC],DL          ;SAVE CURRENT SECTOR
        MOV     CX,WORD PTR [DI].HDLIM ;GET NUMBER OF HEADS
        XOR     DX,DX   ;DIVIDE TRACKS BY HEADS PER CYLINDER
        DIV     CX
        MOV     [CURHD],DL      ;SAVE CURRENT HEAD
        MOV     [CURTRK],AX     ;SAVE CURRENT TRACK
SEEK:
        PUSH    BX              ;Xaddr
        PUSH    DI              ;BPB pointer
        CALL    CHKNEW          ;Unload head if change drives
        CALL    DRIVESEL
        MOV     BL,[DRIVE]
        XOR     BH,BH           ;BX drive index
        ADD     BX,OFFSET TRKTAB        ;Get current track
        MOV     AX,[CURTRK]
        MOV     DL,AL          ;Save desired track
        XCHG    AL,DS:[BX]     ;Make desired track current
        OUT     DISK+1,AL      ;Tell Controller current track
        CMP     AL,DL          ;At correct track?
        JZ      SEEKRET        ;Done if yes
        MOV     BH,2           ;Seek retry count
        CMP     AL,-1          ;Position Known?
        JNZ     NOHOME         ;If not home head
TRYSK:
        CALL    HOME
        JC      SEEKERR
NOHOME:
        MOV     AL,DL
        OUT     DISK+3,AL       ;Desired track
        MOV     AL,1CH+STPSPD   ;Seek
        CALL    DCOM
        AND     AL,98H    ;Accept not rdy, seek, & CRC errors
        JZ      SEEKRET
        JS      SEEKERR         ;No retries if not ready
        DEC     BH
        JNZ     TRYSK
SEEKERR:
        MOV     BL,[DRIVE]
        XOR     BH,BH           ;BX drive index
        ADD     BX,OFFSET TRKTAB        ;Get current track
        MOV     BYTE PTR DS:[BX],-1     ;Make current track
                                        ;unknown
```

```
        CALL    GETERRCD
        MOV     CX,[SECCNT]     ;Nothing transferred
        POP     BX              ;BPB pointer
        POP     DI              ;Xaddr
        RET

SEEKRET:
        POP     BX              ;BPB pointer
        POP     DI              ;Xaddr
        CLC
        RET

;-----------------------------------------------
;
;       READ
;

DISKRD:
ASSUME  DS:CODE
        MOV     CX,[SECCNT]
RDLP:
        CALL    PRESET
        PUSH    BX
        MOV     BL,10               ;Retry count
        MOV     DX,DISK+3           ;Data port
RDAGN:
        MOV     AL,80H              ;Read command
        CLI                         ;Disable for 1793
        OUT     DISK,AL             ;Output read command
        MOV     BP,DI               ;Save address for retry
        JMP     SHORT RLOOPENTRY
RLOOP:
        STOSB
RLOOPENTRY:
        IN      AL,DISK+5           ;Wait for DRQ or INTRQ
        SHR     AL,1
        IN      AL,DX               ;Read data
        JNC     RLOOP
        STI                         ;Ints OK now
        CALL    GETSTAT
        AND     AL,9CH
        JZ      RDPOP               ;Ok
        MOV     DI,BP               ;Get back transfer
        DEC     BL
        JNZ     RDAGN
        CMP     AL,10H              ;Record not found?
        JNZ     GOT_CODE            ;No
        MOV     AL,1                ;Map it
GOT_CODE:
        CALL    GETERRCD
        POP     BX
        RET
RDPOP:
        POP     BX
        LOOP    RDLP
```

```
        CLC
        RET

;-----------------------------------------
;
;       WRITE
;

DISKWRT:
ASSUME  DS:CODE
        MOV     CX,[SECCNT]
        MOV     SI,DI
        PUSH    ES
        POP     DS
ASSUME  DS:NOTHING
WRLP:
        CALL    PRESET
        PUSH    BX
        MOV     BL,10                   ;Retry count
        MOV     DX,DISK+3               ;Data port
WRAGN:
        MOV     AL,0A0H             ;Write command
        CLI                         ;Disable for 1793
        OUT     DISK,AL             ;Output write command
        MOV     BP,SI               ;Save address for retry
WRLOOP:
        IN      AL,DISK+5
        SHR     AL,1
        LODSB                       ;Get data
        OUT     DX,AL               ;Write data
        JNC     WRLOOP
        STI                         ;Ints OK now
        DEC     SI
        CALL    GETSTAT
        AND     AL,0FCH
        JZ      WRPOP               ;Ok
        MOV     SI,BP               ;Get back transfer
        DEC     BL
        JNZ     WRAGN
        CALL    GETERRCD
        POP     BX
        RET
WRPOP:
        POP     BX
        LOOP    WRLP
        CLC
        RET

PRESET:
ASSUME  DS:NOTHING
        MOV     AL,[CURSEC]
        CMP     AL,CS:[BX].SECLIM
        JBE     GOTSEC
        MOV     DH,[CURHD]
        INC     DH
```

```
        CMP     DH,CS:[BX].HDLIM
        JB      SETHEAD             ;Select new head
        CALL    STEP                ;Go on to next track
        XOR     DH,DH               ;Select head zero
SETHEAD:
        MOV     [CURHD],DH
        CALL    DRIVESEL
        MOV     AL,1                ;First sector
        MOV     [CURSEC],AL         ;Reset CURSEC
GOTSEC:
        OUT     DISK+2,AL       ;Tell controller which sector
        INC     [CURSEC]        ;We go on to next sector
        RET

STEP:
ASSUME  DS:NOTHING
        MOV     AL,58H+STPSPD  ;Step in w/ update, no verify
        CALL    DCOM
        PUSH    BX
        MOV     BL,[DRIVE]
        XOR     BH,BH           ;BX drive index
        ADD     BX,OFFSET TRKTAB        ;Get current track
        INC     BYTE PTR CS:[BX]        ;Next track
        POP     BX
        RET

HOME:
ASSUME  DS:NOTHING
        MOV     BL,3
TRYHOM:
        MOV     AL,0CH+STPSPD   ;Restore with verify
        CALL    DCOM
        AND     AL,98H
        JZ      RET3
        JS      HOMERR          ;No retries if not ready
        PUSH    AX              ;Save real error code
        MOV     AL,58H+STPSPD   ;Step in w/ update no verify
        CALL    DCOM
        DEC     BL
        POP     AX              ;Get back real error code
        JNZ     TRYHOM
HOMERR:
        STC
RET3:   RET

CHKNEW:
ASSUME  DS:NOTHING
        MOV     AL,[DRIVE]      ;Get disk drive number
        MOV     AH,AL
        XCHG    AL,[CURDRV]     ;Make new drive current.
        CMP     AL,AH           ;Changing drives?
        JZ      RET1            ;No
; If changing drives, unload head so the head load delay
;one-shot will fire again. Do it by seeking to the same
;track with the H bit reset.
```

```
        IN      AL,DISK+1       ;Get current track number
        OUT     DISK+3,AL       ;Make it the track to seek
        MOV     AL,10H          ;Seek and unload head

DCOM:
ASSUME  DS:NOTHING
        OUT     DISK,AL
        PUSH    AX
        AAM                     ;Delay 10 microseconds
        POP     AX
GETSTAT:
        IN      AL,DISK+4
        TEST    AL,DONEBIT
        JZ      GETSTAT
        IN      AL,DISK
RET1:   RET

DRIVESEL:
ASSUME  DS:NOTHING
;Select the drive based on current info
;Only AL altered
        MOV     AL,[DRIVE]
        OR      AL,SMALBIT + DDBIT      ;5 1/4" IBM PC disks
        CMP     [CURHD],0
        JZ      GOTHEAD
        OR      AL,BACKBIT      ;Select side 1
GOTHEAD:
        OUT     DISK+4,AL       ;Select drive and side
        RET
GETERRCD:
ASSUME  DS:NOTHING
        PUSH    CX
        PUSH    ES
        PUSH    DI
        PUSH    CS
        POP     ES              ;Make ES the local segment
        MOV     CS:[LSTERR],AL ;Terminate list w/ error code
        MOV     CX,NUMERR       ;Number of error conditions
        MOV     DI,OFFSET ERRIN ;Point to error conditions
        REPNE   SCASB
        MOV     AL,NUMERR-1[DI] ;Get translation
        STC                     ;Flag error condition
        POP     DI
        POP     ES
        POP     CX
        RET                     ;and return

;*********************************************************
;       BPB FOR AN IBM FLOPPY DISK, VARIOUS PARAMETERS ARE
;       PATCHED BY GETBP TO REFLECT THE TYPE OF MEDIA
;       INSERTED
;       This is a nine sector single side BPB
DRVBPB:
        DW      512             ;Physical sector size in bytes
        DB      1               ;Sectors/allocation unit
```

```
        DW      1               ;Reserved sectors for DOS
        DB      2               ;# of allocation tables
        DW      64              ;Number directory entries
        DW      9*40            ;Number 512-byte sectors
        DB      11111100B       ;Media descriptor
        DW      2               ;Number of FAT sectors
        DW      9               ;Sector limit
        DW      1               ;Head limit


INITAB  DW      DRVBPB                  ;Up to four units
        DW      DRVBPB
        DW      DRVBPB
        DW      DRVBPB

ERRIN:  ;DISK ERRORS RETURNED FROM THE 1793 CONTROLER
        DB      80H             ;NO RESPONSE
        DB      40H             ;Write protect
        DB      20H             ;Write Fault
        DB      10H             ;SEEK error
        DB      8               ;CRC error
        DB      1               ;Mapped from 10H
                                ;(record not found) on READ
LSTERR  DB      0               ;ALL OTHER ERRORS

ERROUT: ;RETURNED ERROR CODES CORRESPONDING TO ABOVE
        DB      2               ;NO RESPONSE
        DB      0               ;WRITE ATTEMPT
                                ;ON WRITE-PROTECT DISK
        DB      0AH             ;WRITE FAULT
        DB      6               ;SEEK FAILURE
        DB      4               ;BAD CRC
        DB      8               ;SECTOR NOT FOUND
        DB      12              ;GENERAL ERROR

DRV$INIT:
;
; Determine number of physical drives by reading CONFIG.SYS
;
ASSUME  DS:CODE
        PUSH    DS
        LDS     SI,[PTRSAV]
ASSUME  DS:NOTHING
        LDS     SI,DWORD PTR [SI.COUNT] ;DS:SI points to
                                        ;CONFIG.SYS
SCAN_LOOP:
        CALL    SCAN_SWITCH
        MOV     AL,CL
        OR      AL,AL
        JZ      SCAN4
        CMP     AL,"s"
        JZ      SCAN4

WERROR: POP     DS
ASSUME  DS:CODE
```

```
        MOV     DX,OFFSET ERRMSG2
WERROR2: MOV    AH,9
        INT     21H
        XOR     AX,AX
        PUSH    AX                      ;No units
        JMP     SHORT ABORT

BADNDRV:
        POP     DS
        MOV     DX,OFFSET ERRMSG1
        JMP     WERROR2

SCAN4:
ASSUME  DS:NOTHING
;BX is number of floppies
        OR      BX,BX
        JZ      BADNDRV                 ;User error
        CMP     BX,4
        JA      BADNDRV                 ;User error
        POP     DS
ASSUME  DS:CODE
        PUSH    BX                      ;Save unit count
ABORT:  LDS     BX,[PTRSAV]
ASSUME  DS:NOTHING
        POP     AX
        MOV     BYTE PTR [BX].MEDIA,AL          ;Unit count
        MOV     [DRVMAX],AL
        MOV     WORD PTR [BX].TRANS,OFFSET DRV$INIT ;SET
                                                ;BREAK ADDRESS
        MOV     [BX].TRANS+2,CS
        MOV     WORD PTR [BX].COUNT,OFFSET INITAB
                                ;SET POINTER TO BPB ARRAY
        MOV     [BX].COUNT+2,CS
        JMP     EXIT
;
; PUT SWITCH IN CL, VALUE IN BX
;
SCAN_SWITCH:
        XOR     BX,BX
        MOV     CX,BX
        LODSB
        CMP     AL,10
        JZ      NUMRET
        CMP     AL,"-"
        JZ      GOT_SWITCH
        CMP     AL,"/"
        JNZ     SCAN_SWITCH
GOT_SWITCH:
        CMP     BYTE PTR [SI+1],":"
        JNZ     TERROR
        LODSB
        OR      AL,20H          ; CONVERT TO LOWER CASE
        MOV     CL,AL           ; GET SWITCH
        LODSB                   ; SKIP ":"
;
```

```
;  GET NUMBER POINTED TO BY [SI]
;
;  WIPES OUT AX,DX ONLY      BX RETURNS NUMBER
;
GETNUM1:LODSB
        SUB     AL,"0"
        JB      CHKRET
        CMP     AL,9
        JA      CHKRET
        CBW
        XCHG    AX,BX
        MOV     DX,10
        MUL     DX
        ADD     BX,AX
        JMP     GETNUM1

CHKRET: ADD     AL,"0"
        CMP     AL," "
        JBE     NUMRET
        CMP     AL,"-"
        JZ      NUMRET
        CMP     AL,"/"
        JZ      NUMRET
TERROR:
        POP     DS              ; GET RID OF RETURN ADDRESS
        JMP     WERROR
NUMRET: DEC     SI
        RET

ERRMSG1 DB      "SMLDRV: Bad number of drives",13,10,"$"
ERRMSG2 DB      "SMLDRV: Invalid parameter",13,10,"$"
CODE    ENDS
        END
```

DOS Guide

Appendixes

ANSI Escape Sequences

Overview

An ANSI escape sequence consists of an **ESC** key followed by a series of characters and numbers. It can be used to define functions to MS-DOS. For example, an escape sequence can allow you to reassign keys, change graphics functions and modes, erase lines or screens, and affect cursor movement.

This appendix explains how the ANSI escape sequences are defined for MS-DOS version 2.0. Examples on using ANSI escape sequences to redefine keys are included at the end of this appendix.

Notes:

1. Spaces appear between the characters of the escape sequence for purposes of readability only. Do not include any spaces when you type these escape sequences.

2. The default value is used when no explicit value or a value of zero is specified.

3. <n> represents a "numeric parameter." This is a decimal number specified with ASCII digits (0 - 9). The maximum value that can be specified is 255.

4. <s> represents a "selective parameter," which is a decimal number that can be used to select a subfunction. Multiple subfunctions may be selected by separating the parameters with semicolons. Up to 16 semicolons may be used at one time. A semicolon not preceded by a number is equivalent to the default value.

Cursor Functions

The following escape sequences affect the cursor position on the screen.

Scrolling Region - Set Margins

ESC [<t> ; <b> ; r

Sets the scrolling region (margins) to the lines specified for the top of the region <t> and the bottom <b>. The region must be two or more lines. Once set, the cursor cannot be moved beyond the margins except with the absolute cursor position command (see below) unless origin mode has been selected (see "Modes of Operation"). If <t> and <b> are not specified or are 0 and 1, they default to the physical top and bottom of the screen. The sequence **ESC[r** effectively clears the margins.

CUP - Absolute Cursor Position

ESC [<l> ; <c> H

HVP - Horizontal and Vertical Position

ESC [<l> ; <c> f

CUP and HVP move the cursor to the position specified by the parameters, where the first parameter <l> specifies the line number and the second parameter <c> specifies the column number. The default value is 1. If no parameters are specified, the cursor is moved to the home position.

If a scrolling region has been specified and origin mode is in effect (see "Scrolling Region" above), the cursor position is relative to the set margins (where the top margin is line 1). If the line <l> is beyond the margin, the cursor does not move. If the column <c> is beyond the margin, it is placed at the end of the (new) line.

CUU - Cursor Up

ESC [<n> A

CUD - Cursor Down

ESC [<n> B

These sequences move the cursor up or down one line without changing columns. The value of <n> determines the number of lines moved. The default value for <n> is 1. If wrap has not been specified, the sequence is ignored when the cursor is already on the top line (for CUU) or bottom line (for CUD). If wrap is in effect, the cursor will wraparound as necessary.

CUF - Cursor Forward

ESC [<n> C

CUB - Cursor Backward

ESC [<n> D

These sequences move the cursor forward or back one column without changing lines. The value of <n> determines the number of columns moved. The default value for <n> is 1. The sequence is ignored when the cursor is already in the far right column (for CUF) or far left column (for CUB) if wrap is not specified. If wrap is in effect, the cursor will wraparound as necessary.

DSR - Device Status Report

ESC [6 n

The console driver outputs a CPR sequence (see below) on receipt of the DSR escape sequence.

CPR - Cursor Position Report (from console driver to system)

ESC [<l> ; <c> R

The CPR sequence reports current cursor position via standard input. The first parameter <l> specifies the current line and the second parameter <c> 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 (see above). If no SCP was specified, the cursor moves to the home position.

Erasing

The following escape sequences affect erase functions for the screen display or various lines.

ED - Erase Display

ESC [<n> J

where <n> may be:

0 - erase to end of screen (default)
1 - erase to beginning of screen
2 - erase entire screen and place the cursor in the home position

Origin or margins do not affect this sequence. 0 and 1 options do not change the cursor position.

EL - Erase Line

ESC [<n> K

where <n> may be:

0 - erase to end of line (default)
1 - erase to beginning of line
2 - erase entire line

The 0 option erases from the cursor to the end of the line (including the cursor position). None of the options affect the cursor position.

Adding or Deleting Lines

Insert Blank Lines

ESC [<n> L

This sequence inserts <n> blank lines in front of the cursor. It does not affect cursor position. <n> defaults to one. The current line and all following lines are scrolled down to make room for the new line(s). Scrolling is affected by set margins.

Delete Lines

ESC [<n> M

This sequence deletes <n> lines starting at the cursor. It does not affect the cursor position. Subsequent lines are scrolled up, overwriting the current line. Blank lines are inserted at the bottom of the screen. <n> defaults to one.

Modes of Operation

The following escape sequences affect screen graphics.

SGR - Set Graphics Rendition

ESC [<s> ; ... ; <s> m

The SGR escape sequence invokes the graphic functions specified in the following table, where <s> is the number of the desired subfunction. The graphic functions remain in effect until the next occurrence of an SGR escape sequence. If no arguments are given, all attributes are cleared.

Parameter	Parameter Subfunction	
0	All Attributes off	
1	Bold on	
4	Underscore on	(monochrome displays only)
5	Blink on	(not available)
7	Reverse Video on	
8	Concealed on	(ISO 6429 standard)*
30	Black foreground	(ISO 6429 standard)*
31	Red foreground	(ISO 6429 standard)*
32	Green foreground	(ISO 6429 standard)*
33	Yellow foreground	(ISO 6429 standard)*
34	Blue foreground	(ISO 6429 standard)*
35	Magenta foreground	(ISO 6429 standard)*
36	Cyan foreground	(ISO 6429 standard)*
37	White foreground	(ISO 6429 standard)*
38	Superscript on	
39	Subscript on	
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)*
100	Highlight on	
101	Strikethrough on	

* Not implemented

SM - Set Mode

ESC [<s> h

RM - Reset Mode

ESC [<s> l

The SM and RM escape sequences change the screen width or type (see notes below). "l" is a lowercase "L." <s> is one of the following parameters:

Parameter	Parameter Subfunction	
0	40 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	
11	Origin mode	* Ignored

Notes:

1. 0 or 1 -- Selects 40-column mode (default). The screen is cleared and all remaining characters are displayed in double-width.

2. 2 or 3 -- Selects 80-column mode. The screen is cleared and an 80-character screen is used.

3. 4, 5, or 6 -- Ignored.

4. 7 -- Autowrap is set with **ESC[7h** to allow the cursor to wrap around the screen to the following line. To cancel the wraparound, use the sequence **ESC[7l.**

5. 11 -- The sequence ESC[11h enables origin mode, which causes absolute cursor positions to become relative to the set margins (if any). To disable origin mode, specify **ESC[11l.**

Keyboard Reassignment

Although not part of the ANSI 3.64-1979 or ISO 6429 standard, the following keyboard reassignments are compatible with these standards:

```
     ESC [ <n> ; <n> ; ... <n> p
or   ESC [ "string" ; p
or   ESC [ <n> ; "string" ; <n> ; <n> ; "string" ; <n> 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), the first and second code make up an extended ASCII redefinition.

Examples:

1. Reassign the Q and q key to the A and a key (and vice versa):

ESC[65;81p	A becomes Q
ESC[97;113p	a becomes q
ESC[81;65p	Q becomes A
ESC[113;97p	q becomes a

2. Reassign the CTRL 0 key to a DIR command followed by a return:

```
ESC[0;68;"dir";13p
```

The 0;68 is the extended ASCII code for the CTRL 0 key; 13 decimal is a carriage return.

Single-Drive Systems

Instructions for Single-Drive Systems

On a single-drive system, you enter the commands as you would on a multi-drive system. You should think of the single-drive system as having two drives (Drive A and Drive B). Instead of A and B representing two physical drives as on the multi-drive system, however, A and B represent diskettes.

If you specify Drive B when the "Drive A diskette" was last used, you are prompted to insert the diskette for Drive B. For example:

```
A> COPY COMMAND.COM B:

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

  1 File(s) copied

A>_
```

If you specify Drive A when the "Drive B diskette" was last used, you are prompted again to change diskettes. This time, MS-DOS prompts you to insert the "Drive A diskette."

The same procedure is used if a command is executed from a batch file. MS-DOS prompts for you to insert the appropriate disk and to press any key before continuing.

Note: The letter displayed in the system prompt represents the default drive where MS-DOS looks to find a file whose name is entered without a drive specifier. The letter in the system prompt does not represent the last diskette used.

For example, assume that A is the default drive. If the last operation performed was DIR B:, MS-DOS believes the "Drive B diskette" is still in the drive. However, the system prompt is still A, because A is still the default drive. If you type DIR, MS-DOS prompts you for the "Drive A diskette" because Drive A is the default drive, and you did not specify another drive in the DIR command.

Disk Errors

Overview

If a disk or device error occurs at any time during a command or program, MS-DOS returns an error message in the following format:

```
<yyy> ERROR 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.

MS-DOS waits for you to enter one of the following responses:

A Abort. Terminate the program requesting the disk read or write.

I Ignore. Ignore the bad sector and pretend the error did not occur.

R Retry. Repeat the operation. This response is to be used when the operator has corrected the error (such as with NOT READY or WRITE PROTECT errors).

Usually, you want to attempt recovery by entering responses in this order:

R (to try again)
A (to terminate program and try a new disk)

One other error message might be related to faulty disk read or write:

FILE ALLOCATION TABLE BAD FOR DRIVE x

This message means that the copy in memory of one of the allocation tables has pointers to nonexistent blocks. Possibly the disk was incorrectly formatted or not formatted before use. If this error persists, the disk is currently unusable and must be formatted prior to use.

DOS Technical Information

MS-DOS Initialization

MS-DOS initialization consists of several steps. Typically, a ROM (Read Only Memory) bootstrap obtains control and reads the boot sector off the disk. The boot sector then reads the following files:

IO.SYS
MSDOS.SYS

Once these files are read, the boot process begins.

The Command Processor

The command processor supplied with MS-DOS (file COMMAND.COM) consists of three parts:

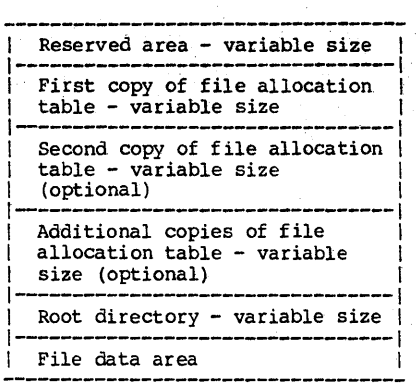
1. **Resident part.** This immediately follows MSDOS.SYS and its data area. This part contains routines to process Interrupts 23H (CTRL C Exit Address) and 24H (Fatal Error Abort Address), as well as a routine which reloads the transient part as required. All standard MS-DOS error handling is performed within this part of COMMAND.COM. This includes displaying error messages and processing the Abort, Retry, or Ignore messages.

2. **Initialization part.** This follows the resident part. During startup, the initialization is given control. Initialization contains the AUTOEXEC file processor setup routine and determines the segment address at which programs can be loaded. It is overlaid by the first program COMMAND.COM loads, since initialization is no longer needed.

3. **Transient part.** This is loaded at the high end of memory and contains all of the internal command processors and the batch file processor. The transient part of the command processor produces the system prompt (such as A>), reads the command from the keyboard or batch file, and causes it to be executed. For external commands, this part builds a command line and issues the EXEC system call (Function Request 4BH) to load and transfer control to the program.

MS-DOS Disk Allocation

The MS-DOS area is formatted as follows:



Assigning space for a file in the data area is not pre-allocated, but is allocated one cluster at a time. A cluster consists of one or more consecutive sectors. All of the clusters for a file are "chained" together in the File Allocation Table (FAT), described in a following section. A second copy of the FAT is kept for error handling. Should the disk develop a bad sector in the middle of the first FAT, the second can be used.

MS-DOS Disk Directory

FORMAT builds the root directory for all disks. Because directories other than the root directory are regarded as files by MS-DOS, there is no limit to the number of files they may contain.

All directory entries are 32 bytes in length and are in the following format (byte offsets are in hexadecimal).

0-7 **Filename.** Eight characters, left-aligned and padded with blanks, if necessary. The first byte of this field indicates the file status as follows:

00H The directory entry has never been used. This is used to limit the length of directory searches to enhance system performance.

2EH The entry is for a directory. If the second byte is also 2EH, the cluster field contains the cluster number of this directory's parent directory (0000H if the parent directory is the root directory). Otherwise, bytes 01H through 0AH are all spaces, and the cluster field contains the cluster number of this directory.

E5H The file was used, but it has been erased.

Any other character is the first character of a filename.

8-0A **Filename extension.**

0B **File attribute.** The attribute byte is mapped as follows (values are in hexadecimal):

01 File is marked read-only. An attempt to open the file for writing using the Open File system call (Function Request 3DH) results in an error code. This value can be used along with other values below. Attempts to delete the file with the Delete File (13H) or Delete a Directory Entry (41H) system call will also fail.

02 Hidden file. The file is excluded from normal directory searches.

04 System file. The file is excluded from normal directory searches.

08 The entry contains the volume label in the first 11 bytes. The entry contains no other usable information (except date and time of creation) and may exist only in the root directory.

10 The entry defines a subdirectory and is excluded from normal directory searches.

20 Archive bit. The bit is set to "on" whenever the file has been written to and closed.

Note: The system files (IO.SYS and MSDOS.SYS) are marked as read-only, hidden, and system files. Files can be marked hidden when they are created. Read-only, hidden, system, and archive attributes may be changed through the Change Attributes system call (Function Request 43H).

0C-15 **Reserved.**

16-17 **Time the file was created or last updated.** The hour, minutes, and seconds are mapped into two bytes as follows:

```
Offset 17H
| H | H | H | H | H | M | M | M |
  7               3   2

Offset 16H
| M | M | M | S | S | S | S | S |
        5   4                   0
```

where:

H is the binary number of hours (0-23)
M is the binary number of minutes (0-59)
S is the binary number of two-second increments

18-19 **Date the file was created or last updated.** The year, month, and day are mapped into two bytes as follows:

```
Offset 19H
| Y | Y | Y | Y | Y | Y | Y | M |
  7                       1   0

Offset 18H
| M | M | M | D | D | D | D | D |
        5   4                   0
```

where:

Y is 0-119 (1980-2099)
M is 1-12
D is 1-31

1A-1B **Starting cluster.** The cluster number of the first cluster in the file.

The first cluster for data space on all disks is cluster 002. The cluster number is stored with the least significant byte first.

Details about converting cluster numbers to logical sector numbers are located in a following section.

1C-1F **File size in bytes.** The first word of this four-byte field is the low-order part of the size.

File Allocation Table (FAT)

The following information is included for system programmers who wish to write installable device drivers. This section explains how MS-DOS uses the File Allocation Table (FAT) to convert the clusters of a file to logical sector numbers. The driver is then responsible for locating the logical sector on disk. Programs must use the DOS file management function calls for accessing files. Programs that access the FAT are not guaranteed to be upwardly compatible with future releases of MS-DOS.

The File Allocation Table is an array of 12-bit entries (1.5 bytes) for each cluster on the disk. The first two FAT entries indicate the size and format of the disk. The second and third bytes always contain FFH.

The third FAT entry, which starts at byte offset 4, begins the mapping of the data area (cluster 002). Files in the data area are not always written sequentially on the disk. The data area is allocated one cluster at a time, skipping over clusters already allocated. The first free cluster found is the next cluster allocated, regardless of its physical location on the disk. This permits the most efficient utilization of disk space, because clusters made available by erasing files can be allocated for new files.

Each FAT entry contains three hexadecimal characters:

000 If the cluster is unused and available.

FF7 The cluster has a bad sector in it. DOS will not allocate such a cluster. CHKDSK counts the number of bad clusters for its report. These bad clusters are not part of any allocation chain.

FF8-FFF Indicates the last cluster of a file.

xxx Any other characters are the cluster number of the next cluster in the file. The cluster number of the first cluster in the file is kept in the file's directory entry.

The File Allocation Table always begins on the first section after the reserved sectors. If the FAT is larger than one sector, the sectors are contiguous. Two copies of the FAT usually are written for data integrity. The FAT is read into one of the DOS buffers whenever needed (open, read, write, etc.). To enhance performance, this buffer is given a high priority to keep it in memory as long as possible.

How to Use the File Allocation Table

Use the directory entry to find the starting cluster of the file. Next, locate each subsequent cluster of the file as shown:

1. Multiply the cluster number just used by 1.5 (each FAT entry is 1.5 bytes long).

2. The whole part of the product is an offset into the FAT, pointing to the entry that maps the cluster just used. That entry contains the cluster number of the next cluster of the file.

3. Use a MOV instruction to move the word at the calculated FAT offset into a register.

4. If the last cluster used was an even number, keep the low-order 12 bits of the register by ANDing it with FFF; otherwise, keep the high-order 12 bits by shifting the register right 4 bits with a SHR instruction.

5. If the resultant 12 bits are FF8H-FFFH, the file contains no more clusters. Otherwise, the 12 bits contain the cluster number of the next cluster in the file.

To convert the cluster to a logical sector number (relative sector, such as that used by Interrupts 25H, 26H and DEBUG):

1. Subtract 2 from the cluster number.

2. Multiply the result by the number of sectors per cluster.

3. Add to this result the logical sector number of the beginning of the data area.

MS-DOS Standard Disk Formats

On an MS-DOS disk, the clusters are arranged on disk to minimize head movement for multi-sided media. All of the space on a track (or cylinder) is allocated before moving on to the next track. This is accomplished by using the sequential sectors on the lowest-numbered head, then all the sectors on the next head, and so on until all sectors on all heads of the track are used. The next sector to be used is sector 1 on head 0 of the next track.

For 5-1/4" disks, the following table can be used:

# of Sides	Sectors/ Track	FAT size Sectors	Dir Sectors	Dir Entries	Sectors/ Cluster
1	8	1	4	64	1
2	8	1	7	112	2
1	9	2	4	64	1
2	9	2	7	112	2

The first byte of the FAT can sometimes be used to determine the format of the disk. The following 5-1/4" formats have been defined for the IBM Personal Computer, based on values of the first byte of the FAT. The formats the following table are the standard disk formats for MS-DOS.

MS-DOS Standard Disk Formats

	5-1/4	5-1/4	5-1/4	5-1/4	8	8	8
No. sides	1	1	2	2	1	1	2
Tracks/side	40	40	40	40	77	77	77
Bytes/sector	512	512	512	512	128	128	1024
Sectors/track	8	9	8	9	26	26	8
Sectors/A.U.	1	1	2	2	4	4	1
Reserved sectors	1	1	1	1	1	4	1
No. FATs	2	2	2	2	2	2	2
Root directory entries	64	64	112	112	68	68	192
No. sectors	320	360	640	720	2002	2002	616
Media Descriptor Byte	FE	FC	FF	FD	FE*	FD	FE*
Sectors for 1 FAT	1	2	1	2	6	6	2

Two media descriptor bytes are the same for 8" disks (FEH). To establish whether a disk is single- or double-density, a read of a single-density address mark should be made. If an error occurs, the media is double-density.

MS-DOS Control Blocks and Work Areas

Overview

This appendix describes the layout of the first 256 bytes of memory, called the Program Segment Prefix. This area is a control block which DOS uses to maintain the system environment.

Typical MS-DOS Memory Map

0000:0000	Interrupt vector table
0070:0000	IO.SYS -- MS-DOS interface to hardware
XXXX:0000	MSDOS.SYS -- MS-DOS interrupt handlers, service routines (Interrupt 21H functions)
	MS-DOS buffers, control areas, and installed device drivers
XXXX:0000	Resident part of COMMAND.COM -- Interrupt handlers for Interrupts 22H (Terminate Address), 23H (CTRL C Exit Address), 24H (Fatal Error Abort Address), and code to reload the transient part
XXXX:0000	External command or utility -- (.COM or .EXE file)
XXXX:0000	User stack for .COM files (256 bytes)
XXXX:0000	Transient part of COMMAND.COM -- Command interpreter, internal commands, batch processor

1. Memory map addresses are in segment:offset format. For example, 0090:0000 is absolute address 0900H.

2. User memory is allocated from the lowest end of available memory that meets the allocation request.

MS-DOS Program Segment

When you type an external command or execute a program through the EXEC system call, DOS determines the lowest available free memory address to use as the start of the program. This area is called the Program Segment.

The first 256 bytes of the Program Segment are set up by the EXEC system call for the program being loaded into memory. The program is then loaded following this block. An .EXE file with minalloc and maxalloc both set to zero is loaded as high as possible.

At offset 0 within the Program Segment, MS-DOS builds the Program Segment Prefix control block. The program returns from EXEC by one of four methods:

1. A long call to location 50H in the Program Segment Prefix with AH=0 or Function Request 4CH.

2. A long jump to offset 0 in the Program Segment Prefix.

3. Issuing an INT 20H with CS:0 pointing at the PSP.

4. Issuing an INT 21H with register AH=0 with CS:0 pointing at the PSP, or 4CH and no restrictions on CS.

Note: It is the responsibility of all programs to ensure that the CS register contains the segment address of the Program Segment Prefix when terminating through any of these methods except for Function Request 4CH. For this reason, using Function Request 4CH is the preferred method.

All four methods result in transferring control to the program that issued the EXEC. During this process, Interrupt addresses 22H (Terminate Address), 23H (CTRL C Exit Address), and 24H (Fatal Error Abort Address) are restored from the values saved in the Program Segment of the terminating program. Control is then given to the Terminate Address. If this is a program returning to COMMAND.COM, control transfers to its resident portion. If a batch file was in process, it is continued. Otherwise, COMMAND.COM performs a checksum on the transient part, reloads it if necessary, issues the system prompt, and waits for you to type the next command.

Conditions for All Programs

When a program receives control, the segment address of the passed environment is contained at offset 2CH in the Program Segment Prefix.

The environment is a series of ASCII strings (totaling less than 32K) in the form:

```
NAME=parameter
```

Each string is terminated by a byte of zeros, and the set of strings is terminated by another byte of zeros. The environment built by the command processor contains at least a COMSPEC= string (the parameters on COMSPEC define the path used by DOS to locate COMMAND.COM on disk). The last PATH and PROMPT commands issued are also in the environment, along with any environment strings defined with the MS-DOS SET command.

The environment that is passed is a copy of the invoking process environment. If your application uses a "keep process" concept, you should be aware that this copy of the environment is static. That is, it will not change even if subsequent SET, PATH, or PROMPT commands are issued.

Offset 50H in the Program Segment Prefix contains code to call the MS-DOS function dispatcher. By placing the desired function request number in AH, a program can issue a far call to offset 50H to invoke an MS-DOS function, instead of issuing an Interrupt 21H. Since this is a call and not an interrupt, MS-DOS may place any code appropriate to making a system call at this position.

The Disk Transfer Address (DTA) is set to 80H (default DTA in the Program Segment Prefix).

File control blocks at 5CH and 6CH are formatted from the first two parameters typed when the command was entered. If either parameter contained a pathname, the corresponding FCB contains only the valid drive number. The filename field is not valid.

The byte at 80H contains the number of characters in the command. An unformatted parameter area at 81H contains all the characters typed after the command (including leading and imbedded delimiters). Redirection of standard input and output is transparent to applications, so redirection command parameters do not appear in this area.

Offset 6 (one word) contains the number of bytes available in the segment.

Register AX indicates whether or not the drive specifiers (entered with the first two parameters) are valid, as follows:

AL=FF if the first parameter contained an invalid drive specifier (otherwise AL=00)

AH=FF if the second parameter contained an invalid drive specifier (otherwise AH=00)

Offset 2 (one word) contains the segment address of the first byte of unavailable memory. Programs must not modify addresses beyond this point unless they were obtained by allocating memory through the Allocate Memory system call (Function Request 48H).

Conditions for Executable (.EXE) Programs

DS and ES registers are set to point to the Program Segment Prefix.

CS, IP, SS, and SP registers are set to the values passed by MS-LINK.

Conditions for Executable (.COM) Programs

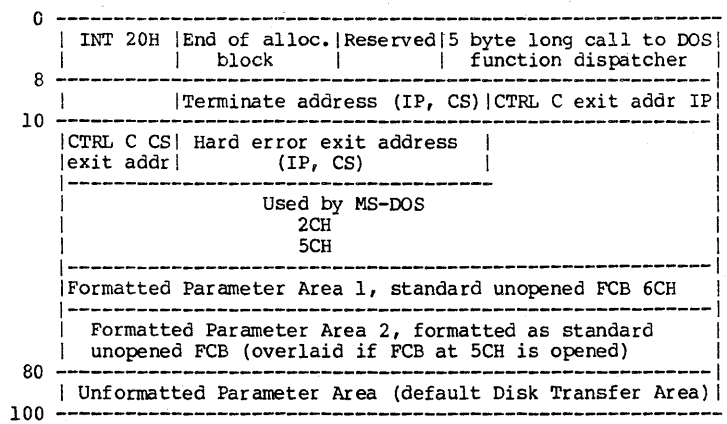
All four segment registers contain the segment address of the initial allocation block which starts with the Program Segment Prefix control block.

All of user memory is allocated to the program. If the program invokes another program through Function Request 4BH, it must first free some memory through the Set Block (4AH) function call to provide space for the program being executed.

The Instruction Pointer (IP) is set to 100H. The Stack Pointer register points to the end of the program's segment. The segment size at offset 6 is reduced by 100H to allow for a stack of that size.

A word of zeros is placed on top of the stack. This allows a user program to exit to COMMAND.COM by doing a RET instruction last. It assumes, however, that the user has maintained his stack and code segments.

The following chart illustrates the format of the Program Segment Prefix. All offsets are in hex. Note that programs must not alter any part of the Program Segment Prefix below offset 5CH.



Miscellaneous Internal Commands

Syntax	Function
BREAK	Display break status
BREAK ON|OFF	Check for CTRL C at keyboard (on or off)
CLS	Clear display screen
CTTY <device>	Change console device
DATE	Display current date
DATE dd-mm-yy	Set new date
EXIT	Exit to a previous command processor
PROMPT	Restore default MS-DOS command prompt d>
PROMPT text	Change command prompt to specified text
SET	Display set values
SET string=string	Set one string equivalent to another
TIME	Display current system time
TIME hh-mm-ss	Set the system time
TYPE <filespec>	Type the contents of the specified file
VER	Display MS-DOS version number
VERIFY ON|OFF	Set verify data function on or off
VOL	Display current volume label
VOL d:	Display label of disk in drive d

Note: <filespec> can include drive, path, file, and extension.

Miscellaneous External Commands

Syntax	Function
CHKDSK d:	Check disk d directory and report status
CHKDSK <filespec> /F	Check disk d and fix errors
CHKDSK <filespec> /V	Check disk d and report with messages
CHKDSK d:>filespec	Redirect CHKDSK output to a file
EXPAND <outfile> <arg1> <filespec> <arg2>	Create several lines in an outfile from argument string(s) and match of filespec
FC <filespec> <filespec>	Compare first file against second file
FC /# <fspec> <fspec>	Specify number of lines needed to match
FC /B <fspec> <fspec>	Force binary comparison of both files
FC /C <fspec> <fspec>	Ignore case of letters during comparison
FC /W <fspec> <fspec>	Compress tabs and spaces in comparison
FC file1 file2 >file3	Redirect comparison output to file3
FIND string	Search screen input for specified string
FIND string <filespec>	Search file(s) for a specified string
FIND /V string	Display lines not containing string
FIND /C string	Print line count of matches in file(s)
FIND /N string	Print relative line number of match(es)
FIXDISK C:	Locate bad sectors on the hard disk
FORMAT d:	Format the specified disk
FORMAT d: /1	Format as a single-sided disk
FORMAT d: /8	Format disk at eight sectors per track
FORMAT d: /O	Format IBM DOS 1.x-compatible disk
FORMAT d: /S	Copy system files after format

Miscellaneous External Commands (continued)

HDFORMAT C:	Format the hard disk
MAKEDB <filespec>	Convert hex values to decimal values
MORE	Display one output screen at a time
PRINT <filespec>	Print a specified file
PRINT <filespec> /T	Terminate (delete) files in print queue
PRINT <filespec> /C	Cancel preceding and following file(s)
PRINT <filespec> /P	Turn on print mode after cancel (/C)
SORT	Sort data alphabetically
SORT /R	Sort alphabetically in reverse (Z to A)
SORT /+n	Sort beginning with column n
SORT <infile >outfile	Put results of sorted infile in outfile
SYS d:	Transfer .SYS files to specified disk

Note: <filespec> can include drive, path, file, and extension.

Batch File Commands

Syntax	Function
ECHO	Display status of batch echo feature
ECHO ON|OFF	Turn batch echo feature on or off
ECHO message	Send a message to the console
FOR %%c IN set DO commd	Repeat command for each %%c in .BAT file
FOR %c IN set DO commd	Repeat command in interactive processing
GOTO label	Transfer control to line after :label
IF [NOT] cond command	Allow conditional execution of command
PAUSE	Suspend execution in a batch file
PAUSE comment	Print comment at execution pause
REM	Insert a blank line in a batch file
REM remark	Display a comment in a batch file
SHIFT	Allow access to over 10 batch parameters

Note: <filespec> can include drive, path, file, and extension.

Directory Commands

Syntax	Function
CHDIR CD -- Synonym	Display current directory
CHDIR ..	Place you in parent directory
CHDIR \	Place you in root directory
CHDIR d:\path	Change directory to path on drive d
DIR	Display default directory
DIR d:	Display directory on drive d
DIR \path	Display directory of specified path
DIR <filespec>	Display specified file(s)
DIR /P	Display one directory screen and pause
DIR /W	Display filenames only, five per line
MKDIR path MD -- Synonym	Make a new directory on default drive
MKDIR d:\path	Make a new directory on drive d
PATH	Display current directory path
PATH d:\	Search root directory on drive d
PATH d:\path	Change path to search named directories
PATH d:\path;d:\path	List of directories to be searched
RMDIR d:\path RD -- Synonym	Remove the specified directory
TREE	Display default directory path(s)
TREE d:	Display directory path(s) on drive d
TREE <filespec>	Search for and display file within tree

Note: <filespec> can include drive, path, file, and extension.

File Management Commands

Syntax	Function
ARCHIVE s: d:/B ARK -- Synonym	Backup files on drive s to drive d
ARCHIVE <filespec> d:/B	Backup specified file(s) to drive d
ARCHIVE <filespec> d:/R	Restore file(s) from drive d
ARCHIVE <filespec> d:/F	Display a directory of file(s)
Note: /x is one of the three options above.	
ARCHIVE s: d:/x /S	Include subdirectories in command
ARCHIVE s: d:/x /M	Include only modified files
ARCHIVE s: d:/x /A	Add archived files to existing disk
ARCHIVE s: d:/x /D:date	Archive or list files after a date
ARCHIVE s: d:/x /P	Prompt if restoring new or R/O files
COPY <file1> <file2>	Copy file1 to file2 on default drive
COPY <fspec> <fspec>	Copy file on drive d to file on drive d
COPY <fspec> <fspec> /V	Copy as above and verify written sectors
COPY CON <filespec>	Copy console input to a specified file
CONVERT	Two-way CP/M to DOS file converter
DEL <filespec> ERASE -- Synonym	Delete specified file on drive d
DISKCOPY s: d:	Copy contents of disk s to disk d
EXE2BIN <filespec>	Convert .EXE file to binary format
EXE2BIN <fspec> <fspec2>	Place EXE2BIN output in filespec2

Note: <filespec> can include drive, path, file, and extension.

File Management Commands (continued)

FILETYPE <filespec>	Display file or directory attributes
FILETYPE <fspec> -x +x	Set or remove file attribute(s): A -- Archive S -- System R -- Read Only <dir> -- Directory H -- Hidden
MOVAFILE <fspec> <fspec>	Transfer files to hard disk or diskette
MOVAFILE <fs> <fs> -A	Transfer file(s) with archive attribute
RECOVER d:	Recover files from disk d
RECOVER <filespec>	Recover only specified file(s)
REN <fspec> <fspec> RENAME -- Synonym	Rename the first file as the second file
SIZE	Display size of all files

CONFIG.SYS File Commands

Syntax	Function
AVAILDEV=[TRUE|FALSE]	Specify available device driver names
BREAK=[ON|OFF]	Check for CTRL C entered at keyboard
BUFFERS=xx	Number of disk buffers in memory
DEVICE=<filespec>	Specify file containing a device driver
FILES=xx	Maximum number of open files accessed by XENIX system calls using DOS handles
SHELL=<filespec>	Specify top-level command processor
SWITCHAR=<char>	Specify device switch designator <char>

Note: <filespec> can include drive, path, file, and extension.

EDLIN Special Editing Keys

Key	Function
CTRL 1	Copies one character from template to new line
CTRL 2 x	Copies all characters from template to new line, up to the specified character
CTRL 3	Copies remaining characters in template to line
DEL	Does not copy (skips over) a character in template
CTRL 4 x	Does not copy (skips over) characters in template, up to the specified character
CTRL X	Voids current input; leaves template unchanged
CTRL DEL	Toggle to enter and exit insert mode
CTRL 5	Makes the new line the new template

Function Requests - Summary

Terminate Program (Function 00H)

Function 00H is called by Interrupt 20H. It causes the current process to terminate and returns control to its parent process. All open file handles are closed and the disk cache is cleaned. This function is often used in terminating old .COM files. The CS register must contain the segment address of the Program Segment Prefix before you call this interrupt. All file buffers are flushed to disk.

Warning: Close all files that have changed in length before calling this function, or their length may not be recorded correctly in the directory.

Read Keyboard and Echo (Function 01H)

Function 01H waits for a character to be typed at the keyboard, then echos the character to the display and returns it in AL. If the character is CTRL-C, Interrupt 23H is executed and all registers are set to the value they had when the original call to MS-DOS was made.

Display Character (Function 02H)

Function 02H displays the character in DL. If CTRL-C is typed, Interrupt 23H is issued.

Auxiliary Input (Function 03H)

Function 03H waits for a character from the auxiliary input device, then returns the character in AL. This system call does not return a status or error code. If a CTRL-C has been typed at console input, Interrupt 23H is issued.

Auxiliary Output (Function 04H)

Function 04H sends the character in DL to the auxiliary output device. This system call does not return a status or error code. If a CTRL-C has been typed at console input, Interrupt 23H is issued.

Print Character (Function 05H)

Function 05H prints the character in DL on the standard printer device. If CTRL-C has been typed at console input, Interrupt 23H is issued.

Direct Console I/O (Function 06H)

The processing depends on the value in DL when the function is called as shown:

DL is FFH (255) -- If a character has been typed at the keyboard, it is returned in AL and the Zero flag is 0; if a character has not been typed, the Zero flag is 1.
DL is not FFH -- The character in DL is displayed.

This function does not check for CTRL-C.

Direct Console Input (Function 07H)

Function 07H waits for a character to be typed, then returns it in AL. This function does not echo the character or check for CTRL-C (see Functions 01H or 08H).

Read Keyboard (Function 08H)

Function 08H waits for a character to be typed, then returns it in AL. If CTRL-C is pressed, Interrupt 23H is executed. This function does not echo the character (see Function 01H).

Display String (Function 09H)

DX must contain the offset (from the segment address in DS) of a string that ends with "$". The string is then displayed (without the "$").

Buffered Keyboard Input (Function 0AH)

DX must contain the offset (from the segment address in DS) of an input buffer in the following form:

Byte:	Contents:
1	Maximum number of characters in the buffer, including the CR (you must set this value).
2	Actual number of characters typed, not counting the CR (the function sets this value).
3-n	Buffer; must be at least as long as the number in byte 1.

This function waits for characters to be typed on the keyboard. These characters are read and placed in the buffer beginning at the third byte until RETURN is typed. If the buffer fills to one less than the maximum, additional characters typed are ignored and ASCII 7 (BEL) is sent to the display until **RETURN** is pressed. The string can be edited as it is being entered.

If **CTRL C** is typed, Interrupt 23H is issued. The second byte of the buffer is set to the number of characters entered (not counting the CR).

Check Keyboard Status (Function 0BH)

This function checks to see if characters are in the type-ahead buffer. If so, AL returns FFH (255); if not, AL returns 0. If CTRL-C is in the buffer, Interrupt 23H is executed.

Flush Buffer, Read Keyboard (Function 0CH)

The keyboard type-ahead buffer is emptied. Further processing depends on the value in AL when the function is called as shown:

1, 6, 7, 8, or 0AH -- The corresponding MS-DOS function is executed.
Any other value -- No further processing; AL returns 0.

Disk Reset (Function 0DH)

Function 0DH is used to ensure that the internal buffer cache matches the disks in the drives. This function writes out modified buffers, marks all buffers in the internal cache as free, and flushes all file buffers. It does not update directory entries; you must close files that have changed to update their directory entries (see Function 10H). This function need not be called before a disk change if all files that changed were closed. It is generally used to force a known state of the system; CTRL-C interrupt handlers should call this function.

Select Disk (Function 0EH)

The drive specified in DL (0 = A:, 1 = B:, etc.) is selected as the default. The number of drives is returned in AL.

Open File (Function 0FH)

DX must contain the offset (from the segment address in DS) of an unopened File Control Block (FCB). The disk directory is searched for the named file. If a directory entry for the file is found, AL returns 0 and the FCB is filled as follows:

-- If the drive code was 0 (default disk), it is changed to the actual disk used (1 = A:, 2 = B:, etc.). This lets you change the default disk without interfering with subsequent operations on this file.

-- The Current Block field (offset 0CH) is set to zero.

-- The Record Size (offset 0EH) is set to the system default (128.)

-- The File Size (offset 10H), Date of Last Write (offset 14H), and Time of Last Write (offset 16H) are set from the directory entry.

Before performing a sequential disk operation on the file, you must set the Current Record field (offset 20H). Before performing a random disk operation on the file, you must set the Relative Record field (offset 21H). Set the default record size to 128 bytes. If a directory entry for the file is not found, AL returns FFH (255).

Close File (Function 10H)

DX must contain the offset (to the segment address in DS) of an opened FCB. The disk directory is searched for the file named in the FCB. This function must be called after a file is changed to update the directory entry. If a directory entry for the file is found, the location of the file is compared with the corresponding entries in the FCB. The directory entry is updated, if necessary, to match the FCB, and AL returns 0. If a directory entry for the file is not found, AL returns FFH (255).

Search for First Entry (Function 11H)

DX must contain the offset (from the segment address in DS) of an unopened FCB. The disk directory is searched for the first matching name. The name can have the wild card character "?" to match any character. To search for hidden or system files, DX must point to the first byte of the extended FCB prefix. If a directory entry for the filename in the FCB is found, AL returns 0 and an unopened FCB of the same type (normal or extended) is created at the Disk Transfer Address. If a directory entry for the filename in the FCB is not found, AL returns FFH (255).

Notes: If an extended FCB is used, the following search pattern is used:

1. If the FCB attribute is zero, only normal file entries are found. Entries for volume label, sub-directories, hidden, and system files will not be returned.

2. If the attribute field is set for hidden or system files or directory entries, it is to be considered as an inclusive search. All normal file entries plus all entries matching the specified attributes are returned. To look at all directory entries except the volume label, the attribute byte may be set to hidden + system + directory (all 3 bits on).

3. If the attribute field is set for the volume label, it is considered an exclusive search and only the volume label entry is returned.

Search for Next Entry (Function 12H)

DX must contain the offset (from the segment address in DS) of an FCB previously specified in a call to Function 11H. Function 12H is used after Function 11H (Search for First Entry) to find additional directory entries that match a filename that contains wild card characters. The disk directory is searched for the next matching name. The name can use the wild card character "?" to match any character. To search for hidden or system files, DX must point to the first byte of the extended FCB prefix. If a directory entry for the filename in the FCB is found, AL returns 0 and an unopened FCB of the same type (normal or extended) is created at the Disk Transfer Address. If a directory entry for the filename in the FCB is not found, AL returns FFH (255).

Delete File (Function 13H)

DX must contain the offset (from the segment address in DS) of an unopened FCB. The directory is searched for a matching filename. The filename in the FCB can contain the wild card character "?" to match any character. If a matching directory entry is found, it is deleted from the directory. If the wild card character "?" is used in the filename, all matching directory entries are deleted and AL returns 0. If no matching directory entry is found, AL returns FFH (255).

Sequential Read (Function 14H)

DX must contain the offset (from the segment address in DS) of an opened FCB. The record pointed to by the current block (offset 0CH) and Current Record (offset 20H) fields is loaded at the Disk Transfer Address, then the Current Block and Current Record fields are incremented. The record size is set to the value at offset 0EH in the FCB. AL returns a code that describes the processing as shown:

Code:	Meaning:
0	Read completed successfully.
1	End-of-file, no data in the record.
2	Not enough room at the Disk Transfer Address to read one record; read cancelled.
3	End-of-file; a partial record was read and padded to the record length with zeros.

Sequential Write (Function 15H)

DX must contain the offset (from the segment address in DS) of an opened FCB. The record pointed to by Current Block (offset 0CH) and Current Record (offset 20H) fields is written from the Disk Transfer Address, then the current block and current record fields are incremented. The record size is set to the value at offset 0EH in the FCB. If the Record Size is less than a sector, the data at the Disk Transfer Address is written to a buffer; the buffer is written to disk when it contains a full sector of data, or the file is closed, or a Reset Disk system call (Function 0DH) is issued. AL returns a code that describes the processing:

Code:	Meaning:
0	Transfer completed successfully.
1	Disk full; write cancelled.
2	Not enough room at the Disk Transfer Address to write one record; write cancelled.

Create File (Function 16H)

DX must contain the offset (from the segment address in DS) of an unopened FCB. The directory is searched for an empty or existing entry for the specified filename. If an empty directory entry is found, it is initialized to a zero-length file, the Open File system call (Function 0FH) is called, and AL returns 0. You can create a hidden file by using an extended FCB with the attribute byte (offset FCB-1) set to 2. If an entry is found for the specified filename, all data in the file is released, making a zero-length file, and the Open File system call (Function 0FH) is issued for the filename (a new, empty file is created). If an empty directory entry is not found and there is no entry for the specified filename, AL returns FFH (255).

Rename File (Function 17H)

DX must contain the offset (from the segment address in DS) of an FCB with the drive number and filename filled in, followed by a second filename at offset 11H. The disk directory is searched for an entry that matches the first filename, which can contain the wild card character "?". If a matching directory entry is found, the filename in the directory entry is changed to match the second filename in the modified FCB (the two filenames cannot be the same name). If the wild card character "?" is used in the second filename, the corresponding characters in the filename of the directory entry are not changed and AL returns 0. If a matching directory entry is not found or an entry is found for the second filename, AL returns FFH (255).

Current Disk (Function 19H)

AL returns the currently selected drive (0 = A:, 1 =B:, etc.).

Set Disk Transfer Address (Function 1AH)

DX must contain the offset (from the segment address in DS) of the Disk Transfer Address. Disk transfers cannot wrap around from the end of the segment to the beginning, nor can they overflow into another segment. If you do not set the Disk Transfer Address, MS-DOS defaults to offset 80H in the Program Segment Prefix.

Random Read (Function 21H)

DX must contain the offset (from the segment address in DS) of an opened FCB. The Current Block (offset 0CH) and Current Record (offset 20H) fields are set to agree with the Relative Record field (offset 21H), then the record addressed by these fields is loaded at the Disk Transfer Address. AL returns a code that describes the processing:

Code:	Meaning:
0	Read completed successfully.
1	End-of-file; no data in the record.
2	Not enough room at the Disk Transfer Address to read one record; read cancelled.
3	End-of-file; a partial record was read and padded to the record length with zeros.

Random Write (Function 22H)

DX must contain the offset from the segment address in DS of an opened FCB. The Current Block (offset 0CH) and Current Record (offset 20H) fields are set to agree with the Relative Record field (offset 21H), then the record addressed by these fields is written from the Disk Transfer Address. If the record size is smaller than a sector (512 bytes), the records are buffered until a sector is ready to write. AL returns a code that describes the processing:

Code:	Meaning:
0	Write completed successfully.
1	Disk is full.
2	Not enough room at the Disk Transfer Address to write one record; write canceled.

File Size (Function 23H)

DX must contain the offset (from the segment address in DS) of an unopened FCB. You must set the Record Size field (offset 0EH) to 128 before calling this function or the function does not return the correct field size. The disk directory is searched for the first matching entry. If a matching directory entry is found, the Relative Record field (offset 21H) is set to the number of records in the file, calculated from the total file size in the directory entry (offset 1CH) and the Record Size field of the FCB (offset 0EH). AL returns 00. If no matching directory is found, AL returns FFH (255).

Set Relative Record (Function 24H)

DX must contain the offset (from the segment address in DS) of an opened FCB. The Relative Record field (offset 21H) is set to the same file address as the Current Block (offset 0CH) and Current Record (offset 20H) fields.

Set Vector (Function 25H)

Function 25H should be used to set a particular interrupt vector so the operating system can manage the interrupts on a per-process basis. (Programs should never set interrupt vectors by writing them directly in the low memory vector table.) DX must contain the offset (to the segment address in DS) of an interrupt-handling routine. AL must contain the number of the interrupt handled by the routine. The address in the vector table for the specified interrupt is set to DS:DX.

Random Block Read (Function 27H)

DX must contain the offset (to the segment address in DS) of an opened FCB. CX must contain the number of records to read; if it contains 0, the function returns without reading any records (no operation). The specified number of records (calculated from the Record Size field at offset 0EH) is read starting at the record specified by the Relative Record field (offset 21H). The records are placed at the Disk Transfer Address. AL returns a code that describes the processing:

Code:	Meaning:
0	Read completed successfully.
1	End-of-file; no data in the record.
2	Not enough room at the Disk Transfer Address to read one record; read canceled.
3	End-of-file; a partial record was read and padded to the record length with zeros.

CX returns the number of records read; the Current Block (offset 0CH), Current Record (offset 20H), and Relative Record (offset 21H) fields are set to address the next record.

Random Block Write (Function 28H)

DX must contain the offset (to the segment address in DS) of an opened FCB; CX must contain either the number of records to write or 0. The specified number of records (calculated from the Record Size field at offset 0EH) is written from the Disk Transfer Address. The records are written to the file starting at the record specified in the Relative Record field (offset 21H) of the FCB. If CX is 0, no records are written, but the File Size field of the directory entry (offset 1CH) is set to the number of records specified by the Relative Record field of the FCB (offset 21H); allocation units are allocated or released, as required. AL returns a code that describes the processing:

Code:	Meaning:
0	Write completed successfully.
1	Disk full. No records written.
2	Not enough room at the Disk Transfer Address to read one record; read canceled.

CX returns the number of records written; the Current Block (offset 0CH), Current Record (offset 20H), and Relative Record (offset 21H) fields are set to address the next record.

Parse File Name (Function 29H)

SI must contain the offset (to the segment address in DS) of a string (command line) to parse; DI must contain the offset (to the segment address in ES) of an unopened FCB. The string is parsed for a filespec (d:filename.ext). If one is found, a corresponding unopened FCB is created at ES:DI. Bits 0-3 of AL control parsing and processing as shown below; 4-7 are ignored.

Bit	Value	Meaning:
0	0	All parsing stops if a file separator is encountered.
	1	Leading separators are ignored.
1	0	The drive number in the FCB is set to 0 (default drive) if the string does not contain a drive number.
	1	The drive number in the FCB is not changed if the string does not contain a drive number.
2	1	The filename in the FCB is not changed if the string does not contain a filename.
	0	The filename in the FCB is set to blanks if the string does not contain a filename.
3	1	The extension in the FCB is not changed if the string does not contain an extension.
	0	The FCB's extension is blank if there is no extension.

If the filename or extension includes an asterisk (*), all remaining characters in the name or extension are set to question mark (?). Filename separators include:

: . ; , = + / " [ ] \ < > | space tab

Filename terminators include all the filename separators plus any control character. A filename cannot contain a filename terminator; if one is encountered, parsing stops. If the string contains a valid filename:

1. AL returns 1 if the filename or extension contains a wild card character (* or ?); AL returns 0 if neither the filename nor extension contains a wild card character.
2. DS:SI point to the first character following the string that was parsed. ES:DI point to the first byte of the unopened FCB.

If the drive letter is invalid, AL returns FFH (255). If the string does not contain a valid filename, ES:DI+1 points to a blank (ASCII 32).

Get Date (Function 2AH)

This function returns the current date set in the operating system as binary numbers in CX and DX:

CX Year (1980-2099)
DH Month (1 = January, 2 = February, etc.)
DL Day (1-31)
AL Day of week (0 = Sunday, 1 = Monday, etc.)

Set Date (Function 2BH)

Registers CX and DX must contain a valid date in binary:

CX Year (1980-2099)
DH Month (1 = January, 2 = February, etc.)
DL Day (1-31)

If the date is valid, the date is set and AL returns 0. If the date is not valid, the function cancels and AL returns FFH (255).

Get Time (Function 2CH)

This function returns the current time set in the operating system as binary numbers in CX and DX:

CH Hour (0-23)
CL Minutes (0-59)
DH Seconds (0-59)
DL Hundredths of a second (0-99)

Set Time (Function 2DH)

Registers CX and DX must contain a valid time in binary:

CH Hour (0-23)
CL Minutes (0-59)
DH Seconds (0-59)
DL Hundredths of a second (0-99)

If the time is valid, the time is set and AL returns 0. If the time is not valid, the function is canceled and AL returns FFH (255).

Set/Reset Verify Flag (Function 2EH)

AL must be either 1 (verify after each disk write) or 0 (write without verifying). MS-DOS checks this flag each time it writes to a disk.

Get Disk Transfer Address (Function 2FH)

Function 2FH returns the DMA transfer address.

Get DOS Version Number (Function 30H)

This function returns the MS-DOS version number. On return, AL.AH will be the two-part version designation (i.e., for DOS 1.28, AL would be 1 and AH would be 28; for pre-1.28, DOS AL = 0). Note that version 1.1 is the same as 1.10, not the same as 1.01.

Keep Process (Function 31H)

This call terminates the current process and attempts to set the initial allocation block to a specific size in paragraphs. It will not free up any other allocation blocks belonging to that process. The exit code passed in AX is retrievable by the parent via Function 4DH. This method is preferred over Interrupt 27H and allows more than 64K to be kept.

CTRL-C Check (Function 33H)

MS-DOS usually checks for a CTRL-C on the controlling device only when doing function call operations 01H-0CH to that device. Function 33H allows the user to expand this checking to include any system call. For example, with the CTRL-C trapping off, all disk I/O proceeds without interruption; with CTRL-C trapping on, the CTRL-C interrupt is given at the system call that initiates the disk operation. Note that programs which use calls 06H or 07H to read CTRL-C's as data must ensure that the CTRL-C check is off.

Get Interrupt Vector (Function 35H)

This function returns the interrupt vector associated with an interrupt. Note that programs should never get an interrupt vector by reading the low memory vector table directly.

Get Disk Free Space (Function 36H)

This function returns information about free space on the disk.

Return Country-Dependent Information (Function 38H)

The value passed in AL is either 0 (for the current country) or a country code. Country codes are typically the international telephone prefix code for the country. If DX = -1, the call sets the current country (as returned by the AL=0 call) to the country code in AL. If the country code is not found, the current country is not changed. **Note:** Applications must assume 32 bytes of information; therefore, the buffer pointed to by DS:DX must be able to accommodate 32 bytes.

Create Sub-Directory (Function 39H)

Given a pointer to an ASCII name, this function creates a new directory entry at the end.

Remove a Directory Entry (Function 3AH)

Function 3AH is given an ASCII name of a directory which is then removed from its parent directory.

Change the Current Directory (Function 3BH)

Function 3BH is given the ASCII name of the directory which is to become the current directory. If any member of the specified pathname does not exist, the current directory is unchanged. Otherwise, the current directory is set to the string.

Create a File (Function 3CH)

Function 3CH creates a new file or truncates an old file to zero length in preparation for writing. If the file did not exist, it is created in the appropriate directory and given the attribute found in CX. The file handle returned has been opened for read/write access.

Open a File (Function 3DH)

Function 3DH associates a 16-bit file handle with a file. The following values are allowed:

Access:	Function:
0	file is opened for reading;
1	file is opened for writing;
2	file is opened for both reading and writing.

DS:DX point to an ASCII name of the file to be opened. The read/write pointer is set at the first byte of the file and the record size of the file is 1 byte. The returned file handle must be used for subsequent I/O to the file.

Close a File Handle (Function 3EH)

If BX is passed a file handle, Function 3EH closes the associated file. Internal buffers are flushed.

Read From File/Device (Function 3FH)

Function 3FH transfers count bytes from a file into a buffer location. All count bytes may not be read; for example, reading from the keyboard will read up to one line of text. If the returned value is zero, the program has tried to read from the end of file. All I/O is done using normalized pointers; no segment wraparound will occur.

Write to a File or Device (Function 40H)

Function 40H transfers count bytes from a buffer into a file. The number of bytes written should be the same as the number requested. The write system call with a count of zero (CX = 0) sets the file size to the current position. Allocation units are allocated or released as required. All I/O is accomplished using normalized pointers; no segment wraparound will occur.

Delete a Directory Entry (Function 41H)

Removes a directory entry associated with a filename.

Move File Pointer (Function 42H)

Function 42H moves the read/write pointer according to one of the following methods:

Method:	Function:
0	The pointer is moved to offset bytes from the beginning of the file.
1	The pointer is moved to the current location plus offset.
2	The pointer is moved to the end of file plus offset.

The offset should be regarded as a 32-bit integer with CX occupying the most significant 16 bits.

Change Attributes (Function 43H)

Given an ASCII name, Function 43H will set or get the attributes of the file to those given in CX. A function code is passed in AL as follows:

0	Return the attributes of the file in CX.
1	Set the attributes of the file to those in CX.

I/O Control for Devices (Function 44H)

Function 44H sets or gets device information associated with an open handle, or sends/receives a control string to a device handle or device. The following values are allowed for the function:

Request:	Function:
0	Get device information (returned in DX)
1	Set device information (as determined by DX)
2	Read CX number of bytes into DS:DX from device control channel.
3	Write CX number of bytes from DS:DX to device control channel.
4	Same as 2 only drive number in BL 0=default,A:=1,B:=2,...
5	Same as 3 only drive number in BL 0=default,A:=1,B:=2,...
6	Get input status
7	Get output status

The status is defined at the time the system is called.

Duplicate a File Handle (Function 45H)

Function 45H takes an already opened file handle and returns a new handle that refers to the same file at the same position.

Force a Duplicate of a Handle (Function 46H)

Function 46H takes an already opened file handle and returns a new handle that refers to the same file at the same position. If a file is already open on handle CX, it is closed first.

Return Text of Current Directory (Function 47H)

Function 47H returns the current directory for a particular drive. The directory is root-relative and does not contain the drive specifier or leading path separator. The drive code passed in DL is 0=default, 1=A:, 2=B:, etc.

Allocate Memory (Function 48H)

Returns a pointer to a free block of memory.

Free Allocated Memory (Function 49H)

Returns a piece of memory to the system pool that was allocated by Function Request 49H.

Modify Allocated Memory Blocks (Function 4AH)

Increases or shrinks an allocated block of memory.

Load and Execute a Program (Function 4BH)

This function allows a program to load another program into memory and begin execution of it. DS:DX points to the ASCII name of the file to be loaded; ES:BX points to a parameter block for the load. A function code is passed in AL as shown:

AL: Function:

0 Load and execute the program. A program header is established for the program and the terminate and CTRL-C addresses are set to the instruction after the EXEC call.
3 Load (do not create) the program header, and do not begin execution. This is useful in loading program overlays.

All open files of a process are duplicated in the child process after an EXEC. An "environment" which conveys various configuration parameters is also inherited from the parent.

Terminate a Process (Function 4CH)

Terminates the current process and transfers control to the invoking process. A return code also may be sent. All files open at the time are closed.

Retrieve the Return Code of a Child (Function 4DH)

Returns the Exit code specified by a child process. It returns this Exit code only once. The low byte of this code is sent by the Exit routine. The high byte is as follows:

0 - Terminate/abort
1 - CTRL-C
2 - Hard error
3 - Terminate and stay resident

Find Match File (Function 4EH)

Uses a pathname with wild card characters in the last component (passed in DS:DX), attributes (passed in CX), and attempts to find all files that match the pathname and have a subset of the required attributes. A datablock at the current DMA is written. To obtain the subsequent matches of the pathname, see Function 4FH.

Step Through a Directory Matching Files (Function 4FH)

Finds the next matching entry in a directory. The current DMA address must point at a block returned by Function 4EH.

Return Current Setting of Verify After Write Flag (Function 54H)

The current value of the verify flag is returned in AL.

Move a Directory Entry (Function 56H)

Renames a file into another path on the same device.

Get/Set Date/Time of File (Function 57H)

Function 57H returns or sets the last-write time for a handle. These times are not recorded until the file is closed. A function code is passed in AL as shown:

AL:	Function:
0	Return the time/date of the handle in CX/DX
1	Set the time/date of the handle to CX/DX

Glossary

.	A notation representing the name of the working directory in a hierarchical directory listing. The working directory is recognized by MS-DOS as the current directory.
..	A notation representing the name of the parent directory in a hierarchical directory listing. The parent directory appears above the working directory in a hierarchical directory listing and immediately precedes the working directory in a PATH command.
Address	A group of binary digits that identify a specific memory location.
Alphanumeric	Referring to a character set that contains letters. numbers. and usually other characters such as punctuation marks.
Ambiguous	Capable of numerous meanings. Therefore, an ambiguous filename is a filename which identifies more than one file.
Append	To add data to the end of an existing file.
ARCHIVE (ARK)	MS-DOS command that backs up and restores files on a hard disk.
ASCII	The American Standard Code for Information Interchange. A seven bit code which represents alphanumeric characters.
Assembler	A program that translates assembly language into machine code that the processor can execute.
AUTOEXEC.BAT	A file containing commands that are automatically executed when MS-DOS is booted.
Available Space	The space on a diskette which is currently unassigned and therefore usable by a new file.
BACKUP	Attache utility program that copies the entire contents of a diskette to another diskette.
BASIC	Beginner's all-purpose symbolic instruction code, a programming language.

Batch File A file containing a series of commands that can be executed by typing the filename without its extension.

Batch Processing Executing a series of commands contained in a batch file.

Baud Rate The number of bits per second transmitted between two electronic devices.

Binary A number system with two as the base and using only the digits zero and one.

BIOS Basic input/output system. The machine-dependent portion of MS-DOS located in the file IO.SYS.

Boot The loading of MS-DOS programs and/or execution of diagnostics that occur when Attache is powered up or reset. Also referred to as a bootstrap.

BREAK MS-DOS command that checks for a **CTRL C** entered at the keyboard.

Buffer A set of memory locations used to compensate for time differences which retains a temporary copy of the data.

Byte The basic unit of computer information, which comprises one character of data.

CHDIR (CD) MS-DOS command that changes directories or displays the working directory.

CHKDSK MS-DOS command that scans the directory of the default or designated drive and checks for consistency.

CLS MS-DOS command that clears the screen.

COBOL Common business-oriented language, a programming language.

Command An instruction to the computer.

COMMAND.COM The MS-DOS command processor program.

Command Mode The operating state (indicated by a prompt) which allows user input of instructions to the computer.

CONVERT Bi-directional CP/M to MS-DOS file converter.

COPY MS-DOS command that copies file(s) as specified.

CPU Central Processing Unit. A single chip that performs data transfer, control, I/O, and logical instructions by executing instructions obtained from memory.

CRT Cathode Ray Tube, which is the display screen.

CTRL Key Key used in conjunction with other keys for activating multi-function commands and operating modes.

CTTY MS-DOS command that changes the console device.

Data A general term for digital information which can be processed by a computer.

DATE MS-DOS command that displays and sets the date.

DEBUG MS-DOS program debugging tool.

Default The value which will be used by the computer if no other value is specified.

DEL (ERASE) MS-DOS command that deletes files as specified.

DIR MS-DOS command that lists requested directory entries.

Directive The part of an assembly source statement which allows the assembler to generate data and values based on specific conditions at assembly time.

Directory A group of related files and subdirectories. A directory may contain program and data files, file information, and the names of subdirectories.

DISKCOPY MS-DOS command that copies entire diskettes.

Diskette Small flexible magnetic disk where computerized information is stored.

Drive The piece of hardware that holds the diskette and transfers information from the diskette to the processing unit and back.

ECHO MS-DOS command that turns the batch file echo feature on or off.

EDLIN MS-DOS line editor for creating source or text files.

EOF End-of-file. A character (CTRL Z) which indicates the end of a file.

EXE2BIN MS-DOS command that converts executable files to binary format.

EXIT MS-DOS command that exits COMMAND.COM and returns to previous command processor.

EXPAND MS-DOS command that creates several lines from one list of arguments.

Expression In assembly, a formula or operation which translates to a numeric value.

Extension The ".XXX" on a filename where XXX can be any non-special ASCII characters which can be used as a means of file type or identification.

External Command MS-DOS command that resides on disk as a program file.

FC MS-DOS file comparison utility.

File A collection of like records stored under a single filename.

FILETYPE MS-DOS command that changes or displays file and directory attributes.

File Type A means of identifying files according to the classification of data which they contain. The file type is the last ".XXX" portion of the filename, where XXX identifies the type, such as ".TXT" for text file.

FIND MS-DOS command that searches files for a specified text string.

FIXDISK MS-DOS command that locates bad sectors on a hard disk.

Flag An indicator, usually a single binary digit, that informs a program when a condition occurs.

FOR Batch command extension for repeated execution of MS-DOS commands.

FORMAT MS-DOS command that formats a diskette for the MS-DOS operating system.

Format A predetermined arrangement of characters which the computer can properly interpret as meaningful information.

FORTRAN Formula translator, a programming language.

GOTO Batch command extension to transfer control to a specified line in a batch file.

HDFORMAT MS-DOS command that formats a hard disk.

Hex An abbreviation for hexadecimal.

Hexadecimal A number system which uses the base 16. numbers 0 - 9, and letters A - F.

IF Batch command extension to allow conditional execution of MS-DOS commands.

Initialize The process of bringing a device or medium to a predetermined entry state. To initialize a diskette is to set up the diskette so that data may be written on it. To initialize a system is to return it to its original state.

Input To enter data into the computer.

Internal Command MS-DOS command that calls a program contained within the MS-DOS command processor.

Kilobyte One thousand units of the basic computer information. which equals one thousand characters of data (one thousand bytes).

LINK MS-DOS command that combines separately produced object modules.

Logged Drive The disk drive (indicated by the MS-DOS prompt) currently containing the diskette from which utilities and files are copied into memory.

Machine Code The binary representation of data which a computer can directly interpret.

MAKEDB MS-DOS command that converts hex file values to decimal values.

Memory Any device that can store logic states such that data can be accessed and retrieved.

Merge The process of joining files to exist as one file.

Microprocessor The hardware component which is capable of interpreting commands and transferring data; functions as the "brains" of the computer.

MKDIR (MD) MS-DOS command that creates a directory.

Mode One of a computer's operating states which provide the means for keyboard multi-functions and other operating state dependent tasks.

Module Individually named section of a program.

MORE MS-DOS command that displays output one screen at a time.

MOVAFILE MS-DOS command that transfers files with the archive attribute set.

MS-DOS Prompt The "A>" or "B>" which appears at the first column of a line on the screen and indicates the currrent logged drive. This character may be changed with the PROMPT command.

Object Code The output of an assembler, executable code.

Operating System The set of programs that run the computer hardware and interpret software commands.

Output To receive information from the computer.

Overwrite To delete and replace the information contained in a file.

Parameter The part of a command line or program line which further defines the specific operation.

Parent Directory The directory above the working directory in a hierarchical directory listing, which precedes the working directory in a PATH command.

Patch Inserting a change to a routine.

PATH MS-DOS command that specifies the location of an external command in a directory.

Pathing A method of specifying the location of files in directories other than the working directory.

PAUSE MS-DOS command that suspends execution in a batch file.

Piping A method of chaining programs together so that the output from one program is the input for another.

Primary Filename A maximum of eight characters which identify a file.

PRINT MS-DOS command that queues and prints text files.

Program A group of computer instructions which. when executed, cause the system to perform a task.

PROMPT MS-DOS command that designates the command prompt.

R/O Read/Only. Data already stored on a diskette marked R/O can be accessed but no new information can be written on it.

R/W Read/Write. Data stored on a diskette marked R/W can be accessed and new data can be written.

RAM Random Access Memory. A storage device into which high or low states can be written (stored) and later retrieved.

Reboot The loading of MS-DOS programs and/or execution of diagnostics that occurs when the **RESET** key is pressed at the same time as the right-hand **SHIFT** key. Also called a cold boot.

Record A collection of related fields. such as fields which describe a single inventory item.

RECOVER MS-DOS command that recovers files from a damaged disk.

Register A short-term digital storage circuit generally capable of storing one byte of data.

REM MS-DOS command that displays a comment in a batch file.

RENAME (REN) MS-DOS command that renames a file.

Reset To return a system or device to its original state.

RMDIR (RD) MS-DOS command that removes a directory.

ROM Read-Only Memory. A storage device from which data can be repeatedly read out. but whose data has been permanently written in.

Root Directory The directory created automatically when a diskette is formatted. The name of the root directory is the backslash character (\).

Scroll The function that "rolls" lines of text or entire "screenfuls" of text up or down on the screen.

SET MS-DOS command that sets one string value equivalent to another.

SHIFT MS-DOS command that allows access to more than 10 batch replaceable parameters.

SIZE MS-DOS command that displays the size of specified files and the total size of all files.

Software The programs that instruct the computer at each step in the accomplishment of a task.

SORT MS-DOS command that sorts data alphabetically, in ascending or descending order.

SYS MS-DOS command that transfers MS-DOS system files to the specified drive.

System Diskette The diskette which contains MS-DOS.

Terminal Mode The operating state emulating a computer terminal that Attache enters when power is turned on.

Text File A file which contains ASCII data.

TIME MS-DOS command that displays and sets the time.

Trace A debugging method that executes one computer instruction at a time and displays the resulting CPU state.

Transient Not a permanent part of memory.

TREE MS-DOS command that displays directory paths.

TYPE MS-DOS command that displays the contents of the specified file.

Unambiguous Defining one specific unit. Therefore, an unambiguous filename is a filename which identifies one specific file.

Unary An arithmetic operator having only one term, positive or negative.

Utility A program which resides on diskette and assists in the operation and maintenance of the computer.

Valet Attache software programs that allow temporary interruption from a program to perform other operations and then return automatically to the interrupted program.

Variable A factor in a command or a program which is subject to change.

VER MS-DOS command that prints the MS-DOS version number.

VERIFY MS-DOS command that verifies that a file has been written to disk correctly.

VOL MS-DOS command that prints the volume identification label.

Wild Card The characters "*" or "?" which indicate to MS-DOS that any character or characters are acceptable.

Working Directory The directory recognized by MS-DOS as the current directory. The working directory may be changed with the CHDIR command.

Write Protect Preventing a diskette or file from being altered by specifying the file or diskette as Read-Only. This is accomplished either by programming parameters or by placing a write protect tab on the notch on the diskette edge.

Index
