# COMMAND

1.0 and later

Command Processor

External

## Purpose

Loads a secondary copy of the MS-DOS default command processor.

## Syntax

COMMAND [*drive*:][*path*] [*device*] [/E:*n*] [/P] [/C *string*]

where:

| | |
|---|---|
| *path* | is the name of the directory to be searched for COMMAND.COM when the transient portion needs to be reloaded; a drive letter can be included with versions 2.0 and later. |
| *device* | is the name of a character device to be used instead of CON for the command processor's input and output (versions 2.0 and later). |
| /E:*n* | is the initial size, in bytes, of the command processor's environment block (160–32768, default = 160) (version 3.2). |
| /P | fixes the newly loaded command processor permanently in memory (versions 2.0 and later). |
| /C *string* | causes the command processor to behave as a transient program and execute the command or program specified by *string* (versions 2.0 and later). |

## Description

The command processor is the module of the operating system that is responsible for issuing prompts to the user, interpreting commands, loading and executing transient application programs, and interpreting batch files. The file COMMAND.COM contains the MS-DOS default command processor, or shell. It is ordinarily loaded from the root directory of the system disk when the system is turned on or restarted, unless the SHELL command is used in the CONFIG.SYS file to specify another command processor or an alternate location for COMMAND.COM.

With versions 1.x, COMMAND.COM is invoked by the COMMAND command in response to a shell prompt or within a batch file. A second copy of the resident portion of COMMAND.COM is loaded and the memory occupied by the original resident portion is lost. The second copy of the transient portion simply overlays the original transient portion. (Versions 1.x of COMMAND support no switches or other parameters and any specified in the command line are ignored.) With versions 2.0 and later, the new copy of COMMAND.COM is loaded *in addition to* the parent command processor and serves as a secondary command processor.

The *path* parameter specifies the location of the COMMAND.COM file that is used to reload the transient part of the command processor if it is overlaid by application programs. If absent, *path* defaults to the root directory of the system (startup) disk.

The *device* parameter allows a character device other than CON to be used by the command processor for input and output. For example, use of AUX as the *device* parameter allows a personal computer to be controlled from a terminal attached to a serial port, instead of from the usual built-in keyboard and memory-mapped video display.

The secondary copy of COMMAND.COM ordinarily remains in memory and serves as the active command processor until an EXIT command is entered. If a /P switch is used with the COMMAND command, the new copy of COMMAND.COM is fixed in memory and the EXIT command is disabled. In such cases, the memory occupied by previously loaded copies of COMMAND.COM is simply lost.

The /E:*n* switch controls the size of the environment block initially allocated for the command processor. The default size of the block is 160 bytes, but the /E:*n* switch allows the initial allocation to be as large as 32768 bytes. This switch is frequently used when *COMMAND.COM* is included in the SHELL command in the CONFIG.SYS file.

When the /C *string* switch is included in the command line, followed by a string designating a command or program name, the new copy of COMMAND.COM carries out the operation specified by *string* and then exits, returning control to its parent command processor or other program. This option allows a batch file to invoke another batch file and then resume its own execution. (If a batch file names another batch file directly without using COMMAND /C *string* as an intermediary, the first batch file is terminated.) Note that when the /C *string* switch is used in combination with other switches, it must be the last switch in the command line.

A secondary copy of COMMAND.COM always inherits a copy of the environment of the command processor or other program that loaded it. Changes made to the new COMMAND.COM's environment with a SET, PROMPT, or PATH command do not affect the environment of any previously loaded program or command processor.

## Examples

To execute the batch file MENU2.BAT from the batch file MENU1.BAT and then resume execution of MENU1.BAT, include the following line in MENU1.BAT:

```
COMMAND /C MENU2
```

To cause COMMAND.COM to be loaded from the directory \SYSTEM on drive C rather than from the root directory and to allocate an initial environment block of 1024 bytes, include the following line in the CONFIG.SYS file:

```
SHELL=C:\SYSTEM\COMMAND.COM C:\SYSTEM /P /E:1024
```

## Messages

**Bad or missing command interpreter**
The file COMMAND.COM is not present in the root directory of the system disk and no SHELL command is present to specify an alternate command processor file or location, or the location specified for COMMAND.COM in a SHELL command is not correct. This message may also be seen if COMMAND.COM is moved from its original location after the system is booted.

**Invalid device**
The character device specified in the command line is not valid or does not exist.

**Invalid environment size specified**
The value supplied with the /E:*n* switch was less than 160 bytes or greater than 32768 bytes.

# COMP

IBM

Compare Files

External

## Purpose

Compares two files or sets of files. This command is available only with PC-DOS.

## Syntax

COMP [*primary*] [*secondary*]

where:

| | |
|---|---|
| *primary* | is the name of the file to be compared against and can be preceded by a drive and/or path; wildcard characters are permitted in the filename. |
| *secondary* | is the name of the file to be compared with *primary* and can be preceded by a drive and/or path; wildcard characters are permitted in the filename. |

## Description

The COMP command compares one file or set of files with another. As each pair of files is compared, the program reports whether the files are identical, different in size, or the same size but different in content.

The *primary* and *secondary* parameters can be any combination of drive, path, and filename, optionally including wildcards to allow sets of files to be compared. (With versions 1.x, using wildcards does not cause multiple file comparisons — only the first secondary file whose name matches the first primary filename is compared.) The *primary* parameter generally designates the specific files to be compared; the *secondary* parameter is usually only a drive and/or path, except when the files being compared have different names or extensions.

If both *primary* and *secondary* are omitted from the command line, the COMP program prompts for them interactively. If *primary* is given as a drive or path only, COMP assumes *.* to be the primary file. If *secondary* is given as a drive or path only, COMP compares all files on that drive or path whose filenames match those of the primary files.

The COMP command is included only with PC-DOS. MS-DOS versions 2.0 and later provide a similar function in the FC command, which also displays the differences between files.

## Examples

To compare the file MYFILE.DAT on the disk in drive A with the file LEDGER.DAT on the disk in drive B, type

```
C>COMP A:MYFILE.DAT B:LEDGER.DAT  <Enter>
```

To compare all the files in the current directory of the disk in drive A with the corresponding files in the current directory of the disk in drive D, type

```
C>COMP A:*.* D:  <Enter>
```

To compare all the files with the extension .ASM in the directory C:\SOURCE with the corresponding files with extension .BAK on the disk in drive B, type

```
C>COMP C:\SOURCE\*.ASM B:*.BAK  <Enter>
```

## Messages

**10 mismatches - ending compare**
The primary and secondary files are the same size but have more than 10 internal differences. The compare operation on this pair of files is aborted and COMP proceeds to the next pair of files, if any.

***filename* and *filename***
This informational message shows the full filenames of the two files currently being compared.

**Access Denied**
An attempt was made to compare a locked file.

**Cannot compare file to itself**
An attempt was made to compare a file with itself.

**Compare error at OFFSET *nn***
**File 1 = *nn***
**File 2 = *nn***
This informational message itemizes the first 10 differences in data between the two files being compared (if the files are the same size), displaying the file offset and the differing bytes from each file as hexadecimal values.

**Compare more files (Y/N)?**
After all specified pairs of files have been compared, the COMP program allows the entry of another pair of file specifications. Respond with *Y* or press Enter to continue; respond with *N* to terminate the COMP program.

**Enter 2nd file name or drive id**
If the secondary filename was not specified in the COMP command, this message prompts the user to enter it (or a path, if the secondary file has the same name as the primary file).

**Enter primary file name**
If no parameter was entered after COMP, this message prompts the user to enter the primary filename. If a drive or path is specified, COMP assumes *.* for the primary filename.

**EOF mark not found**
The last byte at the logical end of the file was not a Control-Z character (^Z, or 1AH). This message is commonly seen during comparison of two files that are not ASCII text files, such as executable program files.

**Files compare OK**
The files being compared were the same length and contained identical data.

**File not found**
The specified filename was invalid or the file does not exist.

**Files are different sizes**
The two files being compared have different sizes recorded in the directory. No comparison on the data within the files is attempted.

**File sharing conflict**
COMP is unable to compare the two current files because one of the files is in use by another process.

**Incorrect DOS version**
The version of COMP is not compatible with the version of PC-DOS that is running.

**Insufficient memory**
The available system memory is insufficient to run the COMP program.

**Invalid drive specification**
The drive specification in *primary* or *secondary* is invalid or does not exist.

**Invalid path**
The path or directory in *primary* or *secondary* is invalid or does not exist.

**Too many files open**
No more system file handles are available. Increase the value of the FILES command in the CONFIG.SYS file and restart the system.

# CONFIG.SYS

2.0 and later

System Configuration File

## Purpose

Allows the user to configure the operating system.

## Description

The CONFIG.SYS file is an ASCII text file that MS-DOS processes during initialization (when the system is turned on or restarted). It allows the user to configure certain aspects of the operating system, such as the number of internal disk buffers allocated, the number of files that can be open at one time, the formats for date and currency, and the name and location of the executable file containing the command processor. CONFIG.SYS can also contain commands that extend the system with installable device drivers for terminal emulation, virtual disks or RAMdisks, extended or expanded memory, and other special peripheral devices.

The CONFIG.SYS file can be created or modified with EDLIN or with any other editor or word processor that can produce ordinary ASCII text files (nondocument files) and save them to disk. The CONFIG.SYS file must be in the root directory of the disk that is used to start the operating system in order for it to be processed during system initialization. When changes are made to the CONFIG.SYS file, they do not take effect until the system is restarted.

Commands in the CONFIG.SYS file take the form

*command*[=]*value*

(Note that the equal sign is optional; any other valid MS-DOS separator [semicolon, tab, or space] can be used instead.) The commands supported are

| Command | Action |
|---|---|
| BREAK | Controls extended checking for Control-C. |
| BUFFERS | Specifies the number of internal disk-sector buffers available for use by MS-DOS when reading from or writing to a disk. |
| COUNTRY | Controls date, time, and currency formatting. |
| DEVICE | Specifies the filename of an installable device driver. |
| DRIVPARM | Redefines the default characteristics of the resident MS-DOS block device(s) (version 3.2). |
| FCBS | Specifies the maximum number of simultaneously open file control blocks (versions 3.0 and later). |

*(more)*

| Command | Action |
|---|---|
| FILES | Specifies the maximum number of simultaneously open files controlled by handles. |
| LASTDRIVE | Sets the highest valid drive letter (versions 3.0 and later). |
| SHELL | Specifies the filename (and optionally the drive and/or path) of the system command processor. |
| STACKS | Sets the number and size of stack frames for the system. |

Each of these commands is discussed in detail on the following pages.

## Message

**Unrecognized command in CONFIG.SYS**
A command in the CONFIG.SYS file was misspelled, an invalid parameter was used, or a command was included that is not compatible with the version of MS-DOS that is running. Correct the CONFIG.SYS file and restart the system.

# CONFIG.SYS: BREAK

2.0 and later

Configure Control-C Checking

## Purpose

Sets or clears MS-DOS's internal flag for Control-C checking.

## Syntax

BREAK=ON ¦OFF

## Description

Pressing Ctrl-C or Ctrl-Break while a program is running ordinarily terminates the program, unless the program itself contains instructions that disable MS-DOS's Control-C handling. As a rule, MS-DOS checks the keyboard for a Control-C only when a character is read from or written to a character device (keyboard, screen, printer, or auxiliary port). Therefore, if a program executes for long periods without performing such character I/O, detection of the user's entry of a Control-C may be delayed. The BREAK=ON command causes MS-DOS to also check the keyboard for a Control-C at the time of each system call (which slows the system somewhat); the BREAK=OFF command disables such extended Control-C checking. The default setting for BREAK is off.

Extended Control-C checking can also be enabled or disabled at the command prompt with the interactive form of the BREAK command whenever the system is running.

## Example

To enable extended Control-C checking during MS-DOS disk operations, insert the line

```
BREAK=ON
```

into the CONFIG.SYS file and restart the system.

## Message

**Unrecognized command in CONFIG.SYS**
The setting supplied for the BREAK command was not ON or OFF. Correct the CONFIG.SYS file and restart the system.

# CONFIG.SYS: BUFFERS

2.0 and later

Configure Internal Disk Buffers

## Purpose

Sets the number of MS-DOS's internal disk buffers.

## Syntax

BUFFERS=*nn*

where:

*nn* is the number of buffers (1–99, default = 2; default = 3 for IBM PC/AT and compatibles).

## Description

MS-DOS maintains a set of internal buffers (sometimes referred to as a disk cache) in which it keeps copies of the sectors most recently read from or written to the disk. Whenever a program requests a disk read, MS-DOS first searches the disk buffers to determine whether a copy of the disk sector containing the required data is already present in RAM. If the sector is found, the actual disk access is bypassed. This technique can significantly improve the overall performance of the disk operating system.

By using the BUFFERS command in the CONFIG.SYS file, the user can control the number of buffers in MS-DOS's disk cache. The default number of buffers is 2 for an IBM PC, PC/XT, or compatible and 3 for an IBM PC/AT or compatible. The optimum number of buffers varies, depending in part on the characteristics and types of the system disk drives, the types of application programs used on the system, the number and levels of subdirectories in the file structure, and the amount of RAM in the system.

If the system has only floppy-disk drives, the default setting of 2 buffers is sufficient. If the system includes a fixed disk, increasing the number of buffers to 10 or so typically speeds up overall system operation. Configuring the system for too many buffers, however, can actually degrade the performance of the system.

Increases in the number of buffers should be tailored to the type of application most frequently used. For example, allocation of extra disk buffers will not improve the performance of programs that use primarily sequential file access but may considerably enhance the execution times of programs that perform random access on a relatively small number of disk records (such as the index for a database file). In addition, if the system has many subdirectories organized in several levels, increasing the number of buffers can significantly increase the speed of disk operations.

The ideal number of buffers for a given system is difficult to predict because of the interactions between the access time of the disk, the speed of the central processing unit, and the

RAM requirements and disk access behavior of the mix of application programs. However, a reasonably optimal number of buffers can be quickly estimated experimentally by increasing the number of buffers in increments of five or so, restarting the system, performing some simple timing tests on the most frequently used application programs, and observing at what number of buffers system performance begins to degrade.

## Example

To allocate 20 internal disk buffers, insert the line

```
BUFFERS=20
```

into the CONFIG.SYS file and restart the system.

## Message

**Unrecognized command in CONFIG.SYS**
The value supplied for the BUFFERS command was not a number in the range 1 through 99.

# CONFIG.SYS: COUNTRY

2.1 and later

Set Country Code

## Purpose

Configures MS-DOS's internationalization support for a specific country.

## Syntax

COUNTRY=*nnn*

where:

*nnn* is the international telephone dialing prefix for the country (001–999, default = 001):

| | |
|---|---|
| Australia | 061 |
| Belgium | 032 |
| Denmark | 045 |
| Finland | 358 |
| France | 033 |
| Israel | 972 |
| Italy | 039 |
| Netherlands | 031 |
| Norway | 047 |
| Spain | 034 |
| Sweden | 046 |
| Switzerland | 041 |
| United Kingdom | 044 |
| USA | 001 |
| West Germany | 049 |

***Note:*** In versions 2.x (except 2.0), *nnn* is 01 through 99. Individual computer manufacturers determine the specific codes supported by their versions of MS-DOS.

## Description

The COUNTRY command enables the user to tailor MS-DOS's date, time, and currency displays for a specific country. This capability, termed internationalization support, is achieved through use of a country code that controls the contents of the table MS-DOS uses to format these displays (including numeric separators). (The internationalization table is made available to application programs through Interrupt 21H Function 38H.) Beginning with version 3.0, PC-DOS also supports the COUNTRY command.

## Example

In West Germany, the format for the date is *dd.mm.yy*. To configure MS-DOS to use this date format, insert the line

```
COUNTRY=049
```

into the CONFIG.SYS file and restart the system.

## Message

**Invalid country code**
The specified country code is not supported by the version of MS-DOS that is running.

# CONFIG.SYS: DEVICE

2.0 and later

Install Device Driver

## Purpose

Loads and links an installable device driver into the operating system during initialization.

## Syntax

DEVICE=[*drive*:][*path*]*filename* [*options*]

where:

*filename* is the name of the device-driver file, optionally preceded by a drive and/or path.

*options* specifies any switches or other parameters needed by the device driver; the DEVICE command itself has no switches.

## Description

Device drivers are the modules of the operating system that control the interface between the operating system and peripheral devices such as disk drives, magnetic-tape drives, CRT terminals, and printers.

As supplied, MS-DOS already contains device drivers for the keyboard, video display, serial port, printer, real-time clock, and disk devices. Device drivers for additional peripheral devices can be linked into the operating system by adding a DEVICE command to the CONFIG.SYS file, placing the file containing the device driver on the system startup disk (or at the location specified by the *drive*: and/or *path* parameter), and restarting the computer.

If a drive other than the one containing the system disk is named as the location of the device driver, that drive must either be accessible via the system's default disk driver or be a drive configured with a previous DEVICE command.

Most OEM implementations of version 3.2 provide three installable device drivers: ANSI.SYS, which allows the video display and keyboard to be controlled by ANSI standard escape sequences; DRIVER.SYS, which supports external disk drives; and RAMDRIVE.SYS (VDISK.SYS with PC-DOS), which uses a portion of the machine's RAM to emulate a disk drive. *See* USER COMMANDS: ANSI.SYS; DRIVER.SYS; RAMDRIVE.SYS; VDISK.SYS.

Many manufacturers of add-on products for MS-DOS machines (such as network interfaces or Lotus/Intel/Microsoft Expanded Memory boards) also supply installable device drivers for use with their hardware. For information concerning these drivers, see the product manufacturer's user's manual.

## Examples

To load the ANSI standard console driver, insert the line

```
DEVICE=ANSI.SYS
```

into the CONFIG.SYS file, place the file ANSI.SYS in the root directory of the system disk, and restart the system.

To load the RAMDRIVE.SYS driver located in the \DRIVERS directory on the disk in drive A, configuring it for 1024 KB in extended memory, insert the line

```
DEVICE=A:\DRIVERS\RAMDRIVE.SYS 1024 /E
```

into the CONFIG.SYS file and restart the system.

## Messages

**Bad or missing *filename***
The filename specified in the DEVICE command is invalid or does not exist or the file does not contain a valid MS-DOS installable device driver.

**Sector size too large in file *filename***
The specified installable device driver uses a sector size that is larger than the sector size used by any of the system's default disk drivers. Such a driver cannot be used because MS-DOS's internal disk buffers will not be large enough to hold a sector read from the device.

# CONFIG.SYS: DRIVPARM

3.2

Set Block-Device Parameters

## Purpose

Alters the system's list of characteristics for an existing block device.

## Syntax

DRIVPARM=/D:*n* [/C] [/F:*n*] [/H:*n*] [/N] [/S:*n*] [/T:*n*]

where:

/D:*n* is the drive number (0–255; 0 = A, 1 = B, etc.) and must always be the first switch in the command line.

/C indicates that the device provides door-lock-status support.

/F:*n* is a form-factor index from the following table (default = 2 if the DRIVPARM command is present but this switch is omitted):

| | |
|---|---|
| 0 | 320 KB or 360 KB |
| 1 | 1.2 MB |
| 2 | 720 KB |
| 3 | 8-inch single-density floppy disk |
| 4 | 8-inch double-density floppy disk |
| 5 | Fixed disk |
| 6 | Tape drive |
| 7 | Other |

/H:*n* is the number of read/write heads (1–99).

/N indicates that the block device is not removable.

/S:*n* is the number of sectors per track (1–99).

/T:*n* is the number of tracks per side (1–999).

***Note:*** The DRIVPARM command must not be used to specify device characteristics that the device driver is not capable of supporting.

## Description

Whenever the device driver for a block device such as a disk drive or magnetic-tape drive performs input or output, it refers to an internal table of characteristics for the device that allows it to convert logical addresses to physical addresses. The DRIVPARM command modifies the default MS-DOS values in the table of characteristics for a particular block device during system initialization (when the computer is turned on or restarted). Multiple DRIVPARM commands, each modifying the characteristics of a different block device, can be included in the same CONFIG.SYS file. Any characteristics not specifically altered in

the DRIVPARM command for a particular device retain their original values, except for /F:*n*, which defaults to 2.

DRIVPARM commands that alter the characteristics for block devices controlled by *installable* device drivers must follow the DEVICE command that loads the device driver itself.

## Example

Assume that drive B is a floppy-disk drive originally configured for 40 tracks with 8 sectors per track. To reconfigure the drive to read or write 80 tracks of 9 sectors each, insert the line

```
DRIVPARM=/D:1 /S:9 /T:80
```

into the CONFIG.SYS file and restart the system. For this command to be valid the drive must be capable of supporting these parameters.

## Message

**Unrecognized command in CONFIG.SYS**
An invalid parameter was specified in a DRIVPARM command.

# CONFIG.SYS: FCBS

3.0 and later

Set Maximum Open Files Using File Control Blocks (FCBs)

## Purpose

Configures the maximum number of files that can be open concurrently using file control blocks (FCBs). This command has no practical effect unless either the file-sharing support module SHARE.EXE or networking support has been loaded.

## Syntax

FCBS=*m*,*p*

where:

*m* is the maximum number of files that can be open concurrently using FCBs (1–255, default = 4).

*p* is the number of files opened with FCBs that are protected against automatic closure (0–*m*, default = 0).

## Description

MS-DOS supports two methods of file access: file control blocks and file handles. A file control block is a data structure that stores information about an open file. It resides inside an application program's memory space and is accessed by both MS-DOS and the application. (*See* USER COMMANDS: CONFIG.SYS: FILES for information on file handles.)

In a network environment, a large number of active FCBs or improper use of FCBs by an application can seriously degrade the performance of the network as a whole. Consequently, MS-DOS versions 3.0 and later provide the FCBS command to enable the user to limit the number of files that can be open concurrently using FCBs if either the file-sharing support module SHARE.EXE (*see* USER COMMANDS: SHARE) or network support has been loaded. If an application program attempts to exceed the specified number of files, MS-DOS closes the file with the least recently used FCB.

The *p* parameter in the FCBS command line allows the user to protect files from unilateral closure by MS-DOS. The value of *p* is the number of files, counting from the first file opened using an FCB, that cannot be closed automatically.

If the current value of FCBS is *4,0* (the default) when the file-sharing module SHARE.EXE or network support is loaded, MS-DOS automatically increases the maximum number of files that can be open concurrently to 16 and the number of files protected against automatic closure to 8. (When multiple FCBs refer to the same file, the file is counted only once.)

## Examples

To set the maximum number of files that can be concurrently open using FCBs to 10 and protect none of the FCB-opened files against automatic closure by MS-DOS, insert the line

```
FCBS=10,0
```

into the CONFIG.SYS file and restart the system.

To set the maximum number of files that can be concurrently open using FCBs to 8 but protect the first 4 FCB-opened files against automatic closure by MS-DOS, insert the line

```
FCBS=8,4
```

into the CONFIG.SYS file and restart the system.

## Message

**Unrecognized command in CONFIG.SYS**
An invalid number was specified as one of the parameters in the FCBS command.

# CONFIG.SYS: FILES

2.0 and later

Set Maximum Open Files Using Handles

## Purpose

Configures the maximum number of files and/or devices that can be open concurrently using file handles.

## Syntax

FILES=*n*

where:

*n* is the maximum number of files and devices that can be open concurrently using file handles (8–255, default = 8).

## Description

MS-DOS supports two methods of file access: file handles and file control blocks (FCBs). During initialization, MS-DOS allocates a data structure that holds information about files and/or devices opened with the handle, or extended-file-management, function calls. This structure resides inside the operating system's memory space and is accessed only by MS-DOS. (*See* USER COMMANDS: CONFIG.SYS: FCBS.) The default size of this data structure allows 8 files and/or devices to be open concurrently using the file-handle functions. The FILES command enables the user to change the size of the data structure. (Note that increasing the size of the data structure decreases the amount of RAM available to application programs.)

The FILES command controls the maximum number of files and/or devices opened with handles for *all* active processes in the system combined. The limit on the number of files and/or devices opened for a single process using handles is 20 or the number of entries in the allocated data structure, whichever is less. Five of the 20 possible handles for a given process are automatically assigned to standard input, standard output, standard error, standard auxiliary, and standard list. However, since standard input, standard output, and standard error all default to the same device (CON), only three of the allocated data-structure entries are actually expended. In addition, the preassigned standard device handles for a process can be closed and reused for other files and devices, if necessary.

## Example

To set the maximum number of files and/or devices that can be concurrently open using the handle functions to 20, insert the line

```
FILES=20
```

into the CONFIG.SYS file and restart the system.

## Message

**Unrecognized command in CONFIG.SYS**
An invalid number was specified in the FILES command.

# CONFIG.SYS: LASTDRIVE

3.0 and later

Set Highest Logical Drive

## Purpose

Defines the highest letter that MS-DOS will recognize as a disk-drive code.

## Syntax

LASTDRIVE=*drive*

where:

*drive* is a single letter (A–Z).

## Description

MS-DOS block devices (floppy-disk drives, fixed-disk drives, and magnetic-tape drives) are referred to by logical drive codes consisting of a single letter from A through Z. In most MS-DOS systems, drives A and B are floppy-disk drives, drive C is a fixed disk, and drives D and above are such devices as additional fixed disks, RAMdisks, or network volumes. In some cases, a single physical drive (such as a very large fixed disk) is partitioned into two or more logical drives, each of which is assigned a drive letter.

MS-DOS validates the drive code in a command or filename before carrying out a command. In the default case, MS-DOS recognizes a maximum of five drives (A–E), depending on the total number of default devices and devices incorporated into the system using installable device drivers. (MS-DOS does not consider a drive letter valid unless it refers to a physical or logical device.) The LASTDRIVE command configures MS-DOS to accept additional drive codes, to a total of 26 (A–Z). This also makes it possible to use fictitious drive letters with the SUBST command to assign a drive letter to a subdirectory.

If the letter code for a LASTDRIVE command specifies fewer drives than are physically present in the system (including installed device drivers), MS-DOS uses the actual number of physical drives.

## Example

To configure MS-DOS to recognize a maximum of eight logical disk drives, insert the line

```
LASTDRIVE=H
```

into the CONFIG.SYS file and restart the system.

## Message

**Unrecognized command in CONFIG.SYS**
An illegal value was specified in the LASTDRIVE command.

# CONFIG.SYS: SHELL

2.0 and later

Specify Command Processor

## Purpose

Defines the name and, optionally, the location of the file that contains the operating system's command processor.

## Syntax

SHELL=[*drive*:][*path*]*filename* [*options*]

where:

*filename* is the name of the file containing the command processor, optionally preceded by a drive and/or path.

*options* specifies any switches and other parameters needed by the designated command processor; the SHELL command itself has no switches.

## Description

The command processor, or shell, is the user's interface to the operating system. It is responsible for parsing and carrying out the user's commands, including the loading and execution of other programs from the disk. MS-DOS uses the SHELL command in the CONFIG.SYS file to locate and load the command interpreter for the system during its initialization process.

The default shell for MS-DOS is the file COMMAND.COM. This file is loaded by MS-DOS from the root directory of the system disk if no SHELL command is found in the CONFIG.SYS file or if no CONFIG.SYS file exists.

The most common use of the SHELL command is simply to advise MS-DOS that COMMAND.COM is stored in a location other than the root directory; MS-DOS then sets the COMSPEC variable in the environment block to COMMAND.COM, preceded by the location specified in the SHELL command. (This can be verified by typing the SET command at the command prompt.) Another common use of SHELL is to specify switches or other parameters for COMMAND.COM itself (*see* USER COMMANDS: COMMAND).

## Example

To specify the file VISUAL.COM in the root directory of drive C as the system's command processor, insert the line

```
SHELL=C:\VISUAL.COM
```

into the CONFIG.SYS file and restart the system.

## Message

**Bad or missing command interpreter**
The path or filename in the SHELL command is invalid or the file does not exist.

# CONFIG.SYS: STACKS

3.2

Configure Internal Stacks

## Purpose

Defines the number and size of stacks for system interrupt handlers.

## Syntax

STACKS=*number,size*

where:

*number* is the number of stacks allocated for use by interrupt handlers (8–64, default = 9).

*size* is the size of each stack in bytes (32–512, default = 128).

## Description

Each time certain hardware interrupts occur (02H, 08–0EH, 70H, and 72–77H), MS-DOS version 3.2 switches to an internal stack before transferring control to the handler that will service the interrupt. In the case of nested interrupts, MS-DOS checks to ensure that both interrupts do not get the same stack. After the interrupt has been processed, the stack is released. This protects the stacks owned by application programs or system device drivers from overflowing when several interrupts occur in rapid succession.

The STACKS command configures the number and size of internal stacks available for interrupt handling and thus controls the number of interrupts that can exist only partially processed while still allowing another interrupt to occur.

The *number* parameter sets the number of internal stacks to be allocated; *number* must be in the range 8 through 64. The *size* parameter is the number of bytes allocated per stack frame; *size* must be in the range 32 through 512.

If too many interrupts occur too quickly and the pool of internal stack frames is exhausted, the system halts with the message *Internal Stack Overflow*. Increasing the *number* parameter in the STACKS command usually corrects the problem.

## Example

To configure 10 stacks of 256 bytes each for use by interrupt handlers, insert the line

```
STACKS=10,256
```

into the CONFIG.SYS file and restart the system.

## Message

**Unrecognized command in CONFIG.SYS**
An invalid number was specified in the STACKS command.

# COPY

1.0 and later

Copy File or Device

Internal

## Purpose

Copies one or more files from one disk, directory, or filename to another. Can also copy files to or from character devices.

## Syntax

COPY *source* [/A] [/B] [+*source* [/A] [/B]...] [*destination*] [/A] [/B] [/V]

where:

| | |
|---|---|
| *source* | is the names of the file(s) to be copied, optionally preceded by a drive and/or path; wildcard characters are permitted in filenames. The source can also be a device. |
| *destination* | is the location and, optionally, the name(s) for the copied file(s) and can be preceded by a drive; wildcard characters are permitted in the filename. The destination can also be a device. |
| /A | indicates that the previous file is an ASCII text file. |
| /B | indicates that the previous file is a binary file. |
| /V | performs read-after-write verification of destination file(s). |

## Description

The COPY command copies one or more source files to one or more destination files. When multiple files are copied, the name of each source file is displayed as it is processed. The COPY command can also be used to send the contents of a file to a character device or to copy input from a character device into a file.

The *source* parameter identifies the file or files to be copied. It can consist of any combination of drive, path, and filename or it can be a device name. If a path without a filename is specified, all files in the named directory are copied. Several source files can be concatenated into a single destination file by placing a + operator between their names; if the source filename contains a wildcard but the destination name does not, all the source files are concatenated into the specified destination.

***Warning:*** When multiple source files are concatenated into a destination file with the same name as one of the source files, that filename should be specified as the *first* source file. Otherwise, the contents of the source file will be destroyed before the file is copied.

When a device is specified as the source, it is usually the console (CON), for copying keyboard input to a file or another device. Keyboard input is terminated by pressing Ctrl-Z or F6 (on IBM PCs or compatibles) and then the Enter key.

The *destination* parameter also can consist of any combination of drive, path, and filename or be a device name. Unless the source files are being renamed as part of the operation, *destination* is usually simply a drive and/or path specifying where to place the copied files. If no destination is specified, the source file is copied to a file with the same name in the current directory of the default disk drive; if the source file in this case is itself in the current directory of the current drive, an error message is displayed and the copy operation is aborted. If files are being concatenated and no destination is specified, the source files are copied sequentially into one file in the current directory with the same name as the first source file. If the first source file already exists, the second file and any additional specified files are appended sequentially to the first source file.

The /A and /B switches control the manner in which the COPY command operates on a file. Both switches affect the file specification immediately preceding them and any subsequent file specifications in the command until another /A or /B switch is encountered, at which point the new /A or /B switch takes effect for the file immediately preceding it and for any subsequent files.

The /A switch indicates that a file is an ASCII text file. When the /A switch is applied to a source file, the file is copied up to, but not including, the first Control-Z (^Z) character in the file. When the /A switch is applied to a destination file, a Control-Z character is appended by the COPY command as the last character of the new file.

The /B switch indicates a binary file. When /B is applied to a source file, the exact number of bytes in the original file are copied without regard to Control-Z or any other control characters. When the /B switch is applied to a destination file, no Control-Z character is appended to the newly created file.

The default values for the /A and /B switches for file-to-file copies are /A when source files are being concatenated and /B otherwise. When a file is being copied to or from a character device, the /A switch is the default.

The /V switch causes a read-after-write verification of each block of the destination file. Its effect is equivalent to that of the VERIFY ON command. No comparison is made between the source and destination files—the /V switch simply causes MS-DOS to verify that the destination file has been written correctly.

## Examples

To copy the file REPORT.TXT from the root directory of the disk in drive B to a file named FINAL.RPT in the \WP\DOCS directory on the current drive, type

```
C>COPY B:\REPORT.TXT \WP\DOCS\FINAL.RPT  <Enter>
```

To make a copy of the file A:\V2\SOURCE\MENUMGR.C in the current directory of the current drive, type

```
C>COPY A:\V2\SOURCE\MENUMGR.C  <Enter>
```

To copy all files with the extension .DOC in the current directory of the disk in drive A to files with the same filenames but a .TXT extension in the current directory of the current drive, type

```
C>COPY A:*.DOC *.TXT  <Enter>
```

To combine the files PROLOG.C, MENUMGR.C, and EPILOG.C in the current directory of the current drive into a single file named VISUAL.C in the current directory of the current drive, type

```
C>COPY PROLOG.C+MENUMGR.C+EPILOG.C VISUAL.C  <Enter>
```

To append the files MENUMGR.C and EPILOG.C to an existing file named PROLOG.C in the current directory of the current drive, type

```
C>COPY PROLOG.C+MENUMGR.C+EPILOG.C  <Enter>
```

To copy the file MENUMGR.MAP in the current directory of the current drive to the system printer, type

```
C>COPY MENUMGR.MAP PRN  <Enter>
```

To copy input from the keyboard (CON) to a file named MENU.BAT in the current directory of the current drive, type

```
C>COPY CON MENU.BAT  <Enter>
```

Text subsequently entered from the keyboard is placed into the file MENU.BAT until a Ctrl-Z or F6 is pressed.

To copy all files in the \MEMOS directory on the current drive to the \ARCHIVE directory on the disk in drive B, type

```
C>COPY \MEMOS\*.* B:\ARCHIVE  <Enter>
```

or

```
C>COPY \MEMOS B:\ARCHIVE  <Enter>
```

## Messages

***n* File(s) copied**
This informational message is displayed at the completion of a COPY command and indicates the total number of source files processed.

**Cannot do binary reads from a device**
The COPY command specified a copy from a character device in binary mode. Reenter the command without a /B switch.

**Content of destination lost before copy**
One of the source files specified as a destination file was overwritten prior to completion of the copy. When the destination name is the same as one of the source names, that file should be specified as the first source file.

**File cannot be copied onto itself**
The source directory and filename of a file being copied are the same as the destination directory and filename.

**File not found**
A file specified in the COPY command is invalid or does not exist.

**Invalid directory**
A directory specified in the COPY command is invalid or does not exist.

# CTTY

2.0 and later

Assign Standard Input/Output Device

Internal

## Purpose

Specifies the character device to be used as standard input and output.

## Syntax

CTTY *device*

where:

*device* is the logical character-device name.

## Description

MS-DOS ordinarily uses the computer's built-in keyboard and screen (CON) as standard input and output. The CTTY command allows another character device to be assigned instead.

CTTY allows MS-DOS commands to be issued from a terminal attached to the computer's serial port or from another custom device with a screen and keyboard. Although PRN and NUL are valid MS-DOS device names, they should not be used with this command, as they have no input capability.

Programs that do not use MS-DOS function calls to perform their input and output will not be affected by the CTTY command. Microsoft BASIC is an example of such a program.

## Examples

To use a terminal connected to the serial port as standard input and output for programs, type

```
C>CTTY AUX  <Enter>
```

To reinstate the normal keyboard and video display (CON) as standard input and output for programs, type

```
C>CTTY CON  <Enter>
```

on the currently assigned console device.

## Message

**Invalid device**
The specified device is not a legal character-device name or does not exist in the system.

# DATE

Set Date

1.0 and later

Internal

## Purpose

Sets or displays the system date.

## Syntax

DATE *mm-dd-yy*

or

DATE *mm/dd/yy*

or

DATE *mm.dd.yy* (versions 3.0 and later)

where:

| | |
|---|---|
| *mm* | is the month (1–12). |
| *dd* | is the day (1–31). |
| *yy* | is the year (80–99 or 1980–1999; 80–79 or 1980–2079 with versions 3.0 and later). |

## Description

All computers that run MS-DOS have as part of their hardware configuration a timer, or clock, that maintains the current system date and time. Among other uses, the current date and time are inserted into a file's directory entry when the file is created or modified.

The DATE command allows the user to display or modify the current date that is being maintained by the system's real-time clock. The command is executed automatically by MS-DOS when the system is initialized, unless there is an AUTOEXEC.BAT file on the system disk, in which case DATE is executed only if it is included in the file.

A date entered using the DATE command does not permanently change the system date; the newly entered date will be lost when the system is turned off or reset. On IBM PC/ATs and compatibles, which have a built-in battery-backed clock/calendar, the system setup program (found on the Diagnostics for IBM Personal Computer AT disk or equivalent) must be used to permanently alter the date stored in the machine. On IBM PCs, PC/XTs, and compatibles equipped with add-on cards containing battery-backed clock/calendar circuitry, it is generally necessary to run a time/date installation program (included with the card) when the system is turned on to set the system date and time from the clock/calendar on the card. The DATE command usually has no effect on these card-mounted clock/calendars.

The order of the day, month, and year in the DATE command depends on the country code, which is set with the COUNTRY command in the CONFIG.SYS file. The format shown here is for the USA.

## Examples

To set the system date to October 15, 1987, type

```
C>DATE 10-15-87  <Enter>
```

or

```
C>DATE 10/15/87  <Enter>
```

or

```
C>DATE 10.15.87  <Enter>
```

To display the current system date, type

```
C>DATE  <Enter>
```

and MS-DOS will respond in the form

```
Current date is Thu 10-15-1987
Enter new date (mm-dd-yy):
```

To leave the date unchanged, press the Enter key.

## Messages

**Current date is *day mm-dd-yyyy***
**Enter new date (mm-dd-yy):**
This informational message and prompt are displayed when MS-DOS is started and there is no AUTOEXEC.BAT file on the system disk, when the DATE command is entered alone, or when the DATE command is included in the AUTOEXEC.BAT file.

**Invalid date**
**Enter new date (mm-dd-yy):**
The date entered in the command line or in response to the prompt from the DATE command was not formatted properly or was invalid.

# DEL or ERASE

1.0 and later

Delete File

Internal

## Purpose

Deletes a file or set of files. DEL and ERASE are synonymous.

## Syntax

DEL [*drive*:][*path*]*filename*

or

ERASE [*drive*:][*path*]*filename*

where:

*filename* is the name of the file(s) to be deleted, optionally preceded by a drive and/or path; wildcard characters are permitted in the filename.

## Description

The DEL command marks the directory entry for the specified file as deleted and frees the disk sectors occupied by the file. If the command line ends with *.* or a directory name (including the special directory names . and ..), MS-DOS prompts the user for confirmation before deleting all the files in the current or specified directory. Note that in the case of a directory name, the directory itself is not removed; only the files within it are deleted.

***Warning:*** If the filename specification begins with an * wildcard and the extension is also * (for example, *xyz.*), DEL interprets the specification as *.* and prompts the user for confirmation before deleting all files from the current or specified directory.

## Examples

To delete the file HELLO.C from the current directory on the current drive, type

```
C>DEL HELLO.C  <Enter>
```

To delete all files with the extension .OBJ from the \SOURCE directory on the disk in drive D, type

```
C>DEL D:\SOURCE\*.OBJ  <Enter>
```

To delete all files from the current directory on the current drive, type

```
C>DEL *.*  <Enter>
```

or

```
C>DEL .  <Enter>
```

In this case, MS-DOS will prompt for confirmation that all files should be deleted.

To delete all files from the directory \WORD\LETTERS on the current drive, type

```
C>DEL \WORD\LETTERS  <Enter>
```

Again, MS-DOS will prompt for confirmation that all files should be deleted.

## Messages

**Access denied**
The specified file is read-only. Use the ATTRIB command with the –R switch to remove the file's read-only status.

**Are you sure (Y/N)?**
This message prompts the user for confirmation if the command would delete all files in a directory (if the command line ends with a directory name or *.*). Respond with *Y* to delete all files in the directory; respond with *N* to terminate the command.

**File not found**
The filename in the command is invalid or the file does not exist in the specified directory.

**Invalid directory**
One of the directories named in the file specification is invalid or does not exist.

**Invalid drive specification**
The drive code in the file specification is invalid or the named drive does not exist in the system.

# DIR

1.0 and later

Display Directory

Internal

## Purpose

Displays a list of a directory's files and subdirectories.

## Syntax

DIR [*drive*:][*path*][*filename*] [/P] [/W]

where:

| | |
|---|---|
| *filename* | is the name of the file, optionally preceded by a drive and/or path, whose directory entry is to be displayed; wildcard characters are permitted. |
| /P | causes a pause after each screen page of display. |
| /W | causes a wide display of filenames formatted five across. |

## Description

The DIR command displays information about the files in a directory. It also displays information about the volume name of the disk that contains the directory, the total number of files and subdirectories in the directory, and the amount of free space remaining on the disk.

The normal format of the DIR command's output is

```
 Volume in drive C is HARDDISK
 Directory of  C:\ASM

.            <DIR>      9-19-85   7:09p
..           <DIR>      9-19-85   7:09p
LIB          <DIR>      9-17-86  11:31p
SOURCE       <DIR>      9-17-86  11:31p
AT86     EXE    41146   5-13-85   5:18p
CREF     EXE    15028  10-16-85   4:00a
DEBUG    COM    15552   3-07-85   1:43p
EXE2BIN  EXE     2816   3-07-85   1:43p
EXEMOD   EXE    11034  10-16-85   4:00a
EXEPACK  EXE    10848  10-16-85   4:00a
LIB      EXE    28716  10-16-85   4:00a
LINK     EXE    43988  10-16-85   4:00a
MAKE     EXE    24300  10-16-85   4:00a
MAPSYM   EXE    18026  10-16-85   4:00a
MASM     EXE    85566  10-16-85   4:00a
SYMDEB   EXE    37021  10-16-85   4:00a
T86      EXE    49024  12-06-84   4:03p
       17 File(s)    4022272 bytes free
```

The first line shows the volume label of the disk that contains the directory being displayed; the second line gives the full pathname of the directory. The subsequent lines are

the names of the files and subdirectories within the current or specified directory. Each entry includes the time and date the file or subdirectory was created or last modified.

Files are shown with their exact size in bytes; directories are shown with the symbol <DIR>. If the directory being listed is not the root directory of the disk, it always contains the two special directory entries . and .., which are aliases for the current directory and the parent directory, respectively. These aliases are included in the total file count in the last line of the display.

Subsets of the files and subdirectories in the current or specified directory of the current or specified drive can be listed by including a filename with wildcards in the command line. For example, the filename *.DOC will cause DIR to list only the files with a .DOC extension.

If the command line ends with a drive or path, DIR automatically appends an *.*, causing all files and subdirectories in the current or specified directory of the current or specified drive to be listed. If a filename is included but no extension is given, DIR appends a .* to the filename, causing all files with that name to be listed, regardless of their extension. If a filename ending with a . is included, nothing is appended and all matching subdirectories and filenames without extensions are listed.

The /P switch causes a pause in the display after each screen page (23 lines plus a message). The listing resumes when the user presses a key.

The /W switch causes the list to be in a more compact format by omitting size and date/time information and by displaying the filenames five across:

```
 Volume in drive C is HARDDISK
 Directory of  C:\ASM
.             ..            LIB           SOURCE        AT86     EXE
CREF     EXE  DEBUG    COM  EXE2BIN  EXE  EXEMOD   EXE  EXEPACK  EXE
LIB      EXE  LINK     EXE  MAKE     EXE  MAPSYM   EXE  MASM     EXE
SYMDEB   EXE  T86      EXE
       17 File(s)   4022272 bytes free
```

When the /W form of the listing is displayed, subdirectories are not easily distinguished from files because the <DIR> symbol is not shown.

## Examples

To list all files in the current directory on the current drive, type

```
C>DIR  <Enter>
```

To list all files in the current directory on the disk in drive B, type

```
C>DIR B:  <Enter>
```

or

```
C>DIR B:*.*  <Enter>
```

To list all files in the directory \SOURCE on the current drive, type

```
C>DIR \SOURCE  <Enter>
```

or

```
C>DIR \SOURCE\*.*  <Enter>
```

To list all files with the extension .OBJ in the \LIB directory on the disk in drive D, type

```
C>DIR D:\LIB\*.OBJ  <Enter>
```

To list all files in the parent directory of the current directory on the current drive, type

```
C>DIR ..  <Enter>
```

To list all files in the current directory on the current drive, sorted by filename and extension, type

```
C>DIR ¦ SORT  <Enter>
```

To list all files in the current directory on the current drive, sorted by extension, type

```
c>DIR ¦ SORT /+10  <Enter>
```

The */+10* instructs SORT to sort the directory entries starting at the tenth column, which is the first column of the filename extension.

To list the subdirectories and files without extensions in the current directory, type

```
C>DIR *.  <Enter>
```

To print the directory on an attached printer instead of displaying it on the screen, type

```
C>DIR > PRN  <Enter>
```

To make a copy of the directory in a file called FILES.TXT, type

```
C>DIR > FILES.TXT  <Enter>
```

## Messages

**File not found**
A filename was included in the command line and no matching files were found.

**Invalid directory**
An element of the path included in the command line does not exist.

**Invalid drive specification**
The specified drive is invalid or is not present in the system.

**Strike a key when ready...**
If the DIR command includes the /P switch, the display is suspended after each 23 lines and this message prompts the user to press a key to see the next screenful of entries.

# DISKCOMP

3.2

Compare Floppy Disks

External

## Purpose

Compares two entire floppy disks on a sector-by-sector basis and reports any differences. This command was included with PC-DOS beginning with version 1.0. To compare individual files, *see* USER COMMANDS: COMP; FC.

## Syntax

DISKCOMP [*drive1*:] [*drive2*:] [/1] [/8]

where:

| | |
|---|---|
| *drive1* | is the drive containing the first disk to be compared. |
| *drive2* | is the drive containing the second disk to be compared. |
| /1 | compares only the first sides of the disks. |
| /8 | compares only the first eight sectors of each track. |

## Description

The DISKCOMP command compares the physical sectors of one floppy disk with those of another. The *drive1* and *drive2* parameters designate the drives holding the two disks to be compared; the drives should always be of the same type. If *drive2* is omitted, DISKCOMP uses the current drive. If both *drive1* and *drive2* are omitted or are identical, DISKCOMP performs the comparison using a single drive, prompting the user to swap disks as required.

Ordinarily, DISKCOMP determines the disk format by inspecting the disk in *drive1*. The /1 and /8 switches override this check so that only one side of the disks or only the first eight sectors of each track are compared, regardless of the actual format of the disks.

If all the sectors on all the tracks are identical, DISKCOMP displays the message *Compare OK*. If differences are found, DISKCOMP reports them by issuing a message that includes the numbers of the track and disk side (read/write head) where the differences occur. Because DISKCOMP works at the level of the disks' physical sectors and is ignorant of the control areas and file structures imposed on a disk by MS-DOS, it also reports as errors bad sectors that were marked during the FORMAT process.

When DISKCOMP finishes comparing two disks, it displays a prompt that allows the user to choose between comparing another pair of disks and returning to the MS-DOS command level.

DISKCOMP cannot be used with a network drive or with a drive created or affected by an ASSIGN, JOIN, or SUBST command, nor can it be used with fixed disks.

## Return Codes

0 Compared disks were identical.
1 Differences were found between the compared disks.
2 DISKCOMP was terminated with a Control-C.
3 Bad sector was found on one of the disks being compared.
4 Initialization error was encountered: not enough memory, syntax error in command line, or invalid drive specified in command line.

***Note:*** Return codes are not present in the PC-DOS version of DISKCOMP.

## Examples

To compare the disk in drive A with the disk in drive B, type

```
C>DISKCOMP A: B:  <Enter>
```

To compare two disks using only drive A, type

```
C>DISKCOMP A: A:  <Enter>
```

To compare only the first side of the disk in drive A with the first side of the disk in drive B, type

```
C>DISKCOMP A: B: /1  <Enter>
```

To compare only the first eight sectors of each track on one side of one disk with the first eight sectors of each track on one side of another disk using only drive A, type

```
C>DISKCOMP A: A: /1 /8  <Enter>
```

## Messages

**Cannot DISKCOMP to or from**
**an ASSIGNed or SUBSTed drive**
One of the specified drives has been affected by an ASSIGN or SUBST command.

**Cannot DISKCOMP to or from**
**a network drive**
One of the specified drives is a network device.

**Compare another diskette (Y/N) ?**
This prompt allows comparison of another pair of disks. Respond with *Y* to cause DISKCOMP to prompt for insertion of the next pair of disks to be compared; respond with *N* to exit to MS-DOS.

**Compare error on side *n*, track *n***
A difference was detected between the two disks being compared.

**Compare OK**
The two disks being compared are identical.

**Compare process ended**
The disk comparison was terminated as the result of a fatal error.

**Comparing *n* tracks,**
***n* sectors per track, *n* side(s)**
This informational message specifies the format of the two disks being compared.

**DEVICE Support Not Present**
The disk drive does not support MS-DOS 3.2 device control.

**Drive *X* not ready**
**Make sure a diskette is inserted into**
**the drive and the door is closed**
DISKCOMP was unable to read the disk in the specified drive.

**Drive types or diskette types**
**not compatible**
Single-sided disks cannot be compared with double-sided disks, nor high-density disks with double-density disks.

**FIRST diskette bad or incompatible**
DISKCOMP is unable to determine the format of the first disk.

**Incorrect DOS version**
The version of DISKCOMP is not compatible with the version of MS-DOS that is running.

**Insert diskette with directory that contains**
**COMMAND.COM in drive *X* and strike any key when ready**
If the system was booted from a floppy disk and the system disk was then removed in order to use DISKCOMP, the user must replace the system disk after the compare operation is complete.

**Insert FIRST diskette in drive *X*:**
**Press any key when ready...**
This message prompts the user to insert the first disk of a pair to be compared.

**Insert SECOND diskette in drive *X*:**
**Press any key when ready...**
This message prompts the user to insert the second disk of a pair to be compared.

**Insufficient memory**
The available system memory is insufficient to load and execute the DISKCOMP program.

**Invalid drive specification**
**Specified drive does not exist**
**or is non-removable**
One of the drives specified in the command line is invalid or does not exist.

**Invalid parameter**
**Do not specify filename(s)**
**Command format: DISKCOMP d: d: [/1][/8]**
A syntax error was detected in the command line, usually caused by an incorrect switch.

**SECOND diskette bad or incompatible**
The second disk of a pair to be compared does not have the same format as the first disk or has bad sectors preventing DISKCOMP from determining its format.

**Unrecoverable read error on drive *X*:**
The disk in the specified drive contains an unreadable sector.

# DISKCOPY

2.0 and later

Copy Floppy Disks

External

## Purpose

Performs a sector-by-sector copy of one entire floppy disk to another floppy disk. This command was included with PC-DOS beginning with version 1.0. To copy individual files, *see* USER COMMANDS: COPY.

## Syntax

DISKCOPY [*drive1*:] [*drive2*:] [/1]

where:

| | |
|---|---|
| *drive1* | is the drive containing the disk to be copied. |
| *drive2* | is the drive containing the disk that will become the copy. |
| /1 | copies only the first side of the disk in *drive1* (MS-DOS version 3.2). |

## Description

The DISKCOPY command duplicates a floppy disk, performing the copy on a physical sector-by-sector basis. The *drive1* parameter specifies the location of the disk to be copied (the source disk). The *drive2* parameter specifies the location of the disk that will become the copy (the destination disk). If *drive2* is omitted, the current drive is used as the destination drive; if both *drive1* and *drive2* parameters are omitted or are the same, DISKCOPY performs the copy operation using a single drive, prompting the user to swap the disks as necessary.

DISKCOPY examines the destination disk before writing any information and terminates with an error message if it does not have the same format as the source disk. If the destination disk is not formatted, DISKCOPY formats it with the same format as the source disk, as part of the DISKCOPY operation.

***Note:*** With MS-DOS versions 2.0 through 3.1, the destination disk must be formatted using the FORMAT command before DISKCOPY can be used. All PC-DOS versions of DISKCOPY will automatically format the destination disk, if necessary.

When DISKCOPY finishes copying a disk, it displays a prompt that allows the user to choose between copying another disk and returning to the MS-DOS command level.

Because DISKCOPY creates an exact duplicate of the source disk, any file fragmentation present on the source disk is also present on the destination disk after the DISKCOPY process is complete. To eliminate fragmentation of the source files, they should be copied to the destination disk individually using COPY or XCOPY.

The DISKCOPY command cannot be used with a network drive or with a drive created or affected by an ASSIGN, JOIN, or SUBST command, nor can it be used with fixed disks.

## Return Codes

0 Disk was copied successfully.
1 Nonfatal but unrecoverable read or write error occurred (no Interrupt 24H generated).
2 DISKCOPY was terminated with a Control-C.
3 Fatal error was encountered: unreadable source disk or unformattable destination disk.
4 Initialization error was encountered: not enough memory, syntax error in command line, or invalid drive specified in command line.

***Note:*** Return codes are not present in the PC-DOS version of DISKCOPY.

## Examples

To copy the contents of the disk in drive A to the disk in drive B, type

```
C>DISKCOPY A: B:  <Enter>
```

To copy the contents of the disk in drive A using only one drive, type

```
C>DISKCOPY A: A:  <Enter>
```

To copy only the first side of the disk in drive A to the first side of the disk in drive B, type

```
C>DISKCOPY A: B: /1  <Enter>
```

## Messages

**Cannot DISKCOPY to or from**
**an ASSIGNed or SUBSTed drive**
One of the specified drives has been affected by an ASSIGN or SUBST command.

**Cannot DISKCOPY to or from**
**a network drive**
One of the specified drives is a network device.

**Copy another diskette (Y/N) ?**
This prompt allows copying of another disk. Respond with *Y* to cause DISKCOPY to prompt for insertion of the next set of disks; respond with *N* to exit to MS-DOS.

**Copying *n* tracks**
***n* sectors per track, *n* side(s)**
This informational message specifies the format of the source disk being copied.

**Copy process ended**
The DISKCOPY process has been successfully completed or has been terminated by a fatal error. In the latter case, this message is preceded by another message explaining the error.

**DEVICE Support Not Present**
The disk drive does not support MS-DOS version 3.2 device control.

**Disk error while reading drive *X*:**
**Abort, Retry, Ignore?**
A bad sector was detected on the source disk. This does not necessarily invalidate the disk copy; the bad sector may originally have been detected and flagged by the FORMAT program and therefore not included in any file. One solution is to copy the files individually using the COPY command.

**Drive *X*: not ready**
**Make sure a diskette is inserted into**
**the drive and the door is closed**
DISKCOPY was unable to read the disk in the specified drive.

**Drive types or diskette types**
**not compatible**
Single-sided disks cannot be copied to or from double-sided disks, nor high-density disks to or from double-density disks.

**Formatting while copying**
The destination disk was not previously formatted. It is given the same format as the source disk as part of the DISKCOPY operation (MS-DOS version 3.2).

**Incorrect DOS version**
The version of DISKCOPY is not compatible with the version of MS-DOS that is running.

**Insert diskette with directory that contains**
**COMMAND.COM in drive *X* and strike any key when ready**
If the system was booted from a floppy disk and the system disk was then removed in order to use DISKCOPY, the user must replace the system disk after the copy operation is complete.

**Insert SOURCE diskette in drive *X*:**
**Press any key when ready...**

or

**Insert TARGET diskette in drive *X*:**
**Press any key when ready...**
These messages prompt the user to insert the source and destination disks before beginning the copy operation.

**Insufficient memory**
The available system memory is insufficient to load and execute the DISKCOPY program.

**Invalid drive specification**
**Specified drive does not exist,**
**or is non-removable**
One of the drives specified in the command line is invalid or does not exist. A fixed disk cannot be the source or destination disk for a DISKCOPY operation.

**Invalid parameter**
**Do not specify filename(s)**
**Command Format: DISKCOPY d: d: [/1]**
A syntax error was detected in the command line, usually caused by an incorrect switch or by the use of a filename instead of (or in addition to) a disk drive.

**SOURCE diskette bad or incompatible**

or

**TARGET diskette bad or incompatible**
The source disk could not be read or the destination disk could not be formatted.

**Target diskette is write protected**
The destination disk has a write-protect tab on it.

**Target diskette may be unusable**
Unrecoverable read or write errors were encountered while copying the source disk to the destination disk. The newly copied disk may not be an accurate copy.

**Unrecoverable read error on drive *X*:**
**side *n*, track *n***

or

**Unrecoverable write error on drive *X*:**
**side *n*, track *n***
The disk in the specified drive contained a sector that could not be successfully read or written.

# DRIVER.SYS

3.2

Configurable External-Disk-Drive Driver

External

## Purpose

Installs and configures external disk drives or assigns logical drive letters to existing floppy-disk drives.

## Syntax

DEVICE=DRIVER.SYS /D:*n* [/C] [/F:*n*] [/H:*n*] [/N] [/S:*n*] [/T:*n*]

where:

/D:*n* is the drive number (0–127 for floppy disks, 128–255 for fixed disks) and must always be the first switch in the command line.

/C specifies that door-lock-status support is available.

/F:*n* is the form-factor index for the device (default = 2):

| | |
|---|---|
| 0 | 320/360 KB |
| 1 | 1.2 MB |
| 2 | 720 KB |
| 3 | 8" single-density floppy disk |
| 4 | 8" double-density floppy disk |
| 5 | fixed disk |
| 6 | magnetic-tape drive |
| 7 | other |

/H:*n* is the number of heads supported by the disk drive (1–99).

/N specifies a nonremovable block device.

/S:*n* is the number of sectors per track (1–40).

/T:*n* is the tracks per read/write head (1–999).

## Description

When the computer is turned on or restarted, MS-DOS assigns numbers to all existing internal disk drives. The DRIVER.SYS file — an installable, configurable block-device driver for external disk drives and other mass-storage devices — allows installation of peripheral devices that are not supported by the resident drivers in the MS-DOS BIOS module. DRIVER.SYS can also assign a logical drive letter to an existing disk drive, thus giving the device two drive letters. (This allows such activities as copying files between like media — for example, copying files from one 1.2 MB 5.25-inch disk to another — using the same drive.)

The /D:*n* switch assigns a unit number to the additional disk drive or specifies the number of the existing disk drive that is to be assigned a logical drive letter. (Floppy-disk unit numbers begin at 0; fixed-disk numbers begin at 80H.) For example, if the system contains two floppy-disk drives (0 and 1), an external floppy-disk drive requiring DRIVER.SYS would be assigned the value 2; MS-DOS would then assign that drive the next available drive letter. If the number used with the /D:*n* switch references an existing drive (for example, 0, the first floppy-disk drive), MS-DOS assigns the drive the next available drive letter, allowing the one drive unit to be referenced by two drive letters. The /D:*n* switch is not optional and must precede all other switches in the command line.

The /C, /F:*n*, and /N switches describe characteristics of the disk drive that is being selected for use with DRIVER.SYS. The /C switch is included only if the device has a status line indicating whether the disk in the drive has been changed. (This information is used by the driver to optimize disk accesses to the directory and file allocation table.) If the device does not have a status line, /C will have no effect. The /F:*n* option describes the form-factor index used by the device. The permissible values for *n* are given in the preceding table; the default type is a 720 KB disk. The /N switch indicates that the block device is nonremovable. Access to such devices is more efficient than access to removable media because MS-DOS can eliminate calls to the driver for a media-change check.

The /H:*n*, /S:*n*, and /T:*n* switches describe the physical layout of the recording medium. /H:*n* specifies the number of recording surfaces, or read-write heads, supported by the drive (1–99). /S:*n* is the number of sectors per track (1–40) and /T:*n* is the tracks per side (1–999). (The total number of physical sectors on a given disk is found by multiplying the number of heads by the tracks per side and the sectors per track.)

***Note:*** The values used with these switches must be supported by the device being installed. If DRIVER.SYS is used to assign a logical drive letter to an existing physical device, the values used with the switches must be identical to the characteristics imposed by the default device driver.

## Examples

To install a driver for an external 720 KB disk drive in a system that already has two 5.25-inch floppy-disk drives, insert the line

```
DEVICE=DRIVER.SYS /D:02
```

into the CONFIG.SYS file and restart the system.

Assume that an IBM PC/AT or compatible has three disk drives installed: Drive A is a 1.2 MB 5.25-inch floppy-disk drive; drive B is a 360 KB 5.25-inch floppy-disk drive; drive C is a 30 MB fixed-disk drive. To assign the logical drive letter D to the existing drive A, effectively giving the one drive two drive letters, insert the line

```
DEVICE=DRIVER.SYS /D:0 /F:1 /H:2 /S:15 /T:80 /C
```

into the CONFIG.SYS file and restart the system.

## Messages

**Bad or missing DRIVER.SYS**
The file DRIVER.SYS could not be found in the root or specified directory or has been damaged.

**ERROR - Incorrect DOS version**
The version of DRIVER.SYS is not compatible with the version of MS-DOS that is running.

**ERROR - No drive specified**
The /D:*n* switch was not included in the command line.

**Loaded External Disk Driver for Drive *X***
The device driver has been successfully installed and this message informs the user of the drive letter assigned to the device.

**Sector size too large in file DRIVER.SYS**
DRIVER.SYS uses a sector size that is larger than the sector size used by any of the system's default disk drivers. The driver cannot be used because MS-DOS's internal disk buffers will not be large enough to hold a sector read from the device.

# EDLIN

Line Editor

1.0 and later

External

## Purpose

Creates and changes ASCII text files.

## Syntax

EDLIN [*drive*:][*path*]*filename* [/B]

where:

| | |
|---|---|
| *filename* | is the name of an ASCII text file to be created or edited, optionally preceded by a drive and/or path. |
| /B | causes logical end-of-file marks within the file to be ignored (versions 2.0 and later). |

## Description

The EDLIN program is a simple line-oriented editor that can be used to create or maintain short text files. The user references and edits text by line number; EDLIN displays these numbers for convenience but they do not become part of the file. Each line of the file being edited can be a maximum of 253 characters.

The *filename* parameter specifies a plain ASCII text file; if the file does not already exist, EDLIN creates it. (EDLIN cannot be used on most files created by word-processing programs because such document files have embedded formatting codes and other formatting information that EDLIN cannot interpret.) EDLIN does not assume any extensions; the user must type the complete filename. (EDLIN does not permit editing of a .BAK file.)

If *filename* is a previously existing text file, EDLIN loads lines from the file into memory until the editing buffer is 75 percent full or until a logical end-of-file mark or the physical end of the file is reached. The /B switch forces EDLIN to ignore any logical end-of-file marks (1AH, or Control-Z) the file may contain. If the file is too large for the edit buffer, the Write Lines to Disk (W) and Append Lines from Disk (A) commands are used during the edit session to process the remaining portions of the file.

Once the file is created or loaded into the editing buffer, EDLIN displays its asterisk prompt (*) and the user can begin entering editing commands.

EDLIN commands consist of a single character, in either uppercase or lowercase, usually preceded by one or more line numbers. More than one command can be entered on a single line by separating the commands with semicolons. EDLIN does not execute a command until the Enter key is pressed.

The EDLIN commands are

| Command | Action |
|---|---|
| *linenumber* | Edit line. |
| A | Append lines from disk. |
| C | Copy lines (versions 2.0 and later). |
| D | Delete lines. |
| E | End editing session. |
| I | Insert lines. |
| L | List lines. |
| M | Move lines (versions 2.0 and later). |
| P | Display in pages (versions 2.0 and later). |
| Q | Quit without saving changes. |
| R | Replace text. |
| S | Search for text. |
| T | Transfer another file into the edit buffer (versions 2.0 and later). |
| W | Write lines to disk. |

Each of these commands is discussed in detail in the following pages.

All EDLIN commands that accept a line number or range of line numbers can also recognize the following symbolic references:

| Symbol | Meaning |
|---|---|
| # | The line after the last line in the edit buffer |
| . | The current line |
| +*n* or −*n* | A line number relative to the current line (for example, +5 = five lines past the current line) |

When the user terminates the editing session with the E command, EDLIN gives the new file the same name as the original file and renames the original (unchanged) file with the extension .BAK. Any previous file with the same name and the extension .BAK is lost. When the user terminates the editing session with the Q command, the original filename remains unchanged.

## Example

To edit the file AUTOEXEC.BAT in the root directory of the current drive, type

```
C>EDLIN \AUTOEXEC.BAT  <Enter>
```

## Messages

**Cannot edit .BAK file — rename file**
Files with the extension .BAK cannot be edited with EDLIN. Rename the file or copy it to a file with the same name but a different extension.

**End of input file**
The entire file has been read into memory.

**File is READ-ONLY**
Files marked with the read-only attribute cannot be edited. Remove the read-only attribute with the ATTRIB command or copy the file to a file with a different name.

**File name must be specified**
The command line did not include a filename.

**File not found**
The file named in the command line could not be found or does not exist.

**Incorrect DOS version**
The version of EDLIN is not compatible with the version of MS-DOS that is running.

**Insufficient memory**
Not enough memory is available to carry out the requested command.

**Invalid drive or file name**
The command line included a drive that is invalid or does not exist in the system or the filename is not valid.

**Invalid Parameter**
The command line contained an illegal switch or other invalid parameter.

**New file**
The file named in the command line did not previously exist. The file is created and the edit buffer is emptied.

**Read error in: *filename***
MS-DOS was unable to read the entire file. Run CHKDSK to determine whether the file or disk has been damaged.

# EDLIN: *linenumber*

1.0 and later

Edit Line

## Purpose

Selects a line of text for editing.

## Syntax

*linenumber*

where:

*linenumber* is the number assigned by EDLIN to the text line to be edited (1–65534).

## Description

The command to edit a particular line of text is simply the line's number or one of the special symbols or expressions that evaluate to a line number, followed by the Enter key. EDLIN displays the current contents of the specified line and copies them to a special editing buffer called the template, then moves the cursor to a new line and displays a prompt in the form of the line number followed by a colon and an asterisk. If a line number is not specified (that is, if the Enter key alone is pressed in response to the EDLIN prompt), EDLIN displays the line following the current line and makes it the current line.

The user can change the text of the specified line by simply entering new text followed by a press of the Enter key, leave the text unchanged by pressing Enter alone, or modify the text by using special editing keys to change a portion of the text that has been placed in the template. These editing keys and their actions are

| Key | Action |
|---|---|
| F1 | Copies one character from the template to the new line. |
| F2*char* | Copies all characters up to the specified character from the template to the new line. |
| F3 | Copies all remaining characters in the template to the new line. |
| Del | Does not copy (skips over) one character. |
| F4*char* | Does not copy (skips over) all characters up to the specified character. |
| Esc | Restarts editing for the current line, leaving the template unchanged. |
| Ins | Enters/exits character-insert mode. |
| F5 | Makes the newly edited line the new template. |
| → | Copies one character from the template to the new line. |
| ← | Deletes one character from the new line. |
| Backspace | Deletes one character from the new line. |

***Note:*** Computers that are not IBM-compatible may use a different set of editing keys to perform these actions.

Control characters (those characters with ASCII codes in the range 0–1FH) cannot be inserted into text with the usual Control-key combinations. Instead, the user must press the sequence Ctrl-V, followed by an uppercase character or symbol. For example, Ctrl-C (ASCII code 03H) is entered into text by pressing Ctrl-V followed by a capital C; the Escape character (ASCII code 1BH) is generated by pressing Ctrl-V followed by a left square-bracket character ([).

## Examples

To edit line 4, type

```
*4  <Enter>
```

To edit the line two lines ahead of the current line, type

```
*+2  <Enter>
```

# EDLIN: A

1.0 and later

Append Lines from Disk

## Purpose

Reads lines from the file being edited into the edit buffer.

## Syntax

[*n*]A

where:

*n* is the number of lines to be read from the file.

## Description

If the file being edited is too large to fit into the edit buffer, EDLIN ordinarily reads only enough text to fill 75 percent of the buffer when it opens the file, reserving 25 percent of the buffer for additions and changes to the text. The user must then employ the Write Lines to Disk (W) and Append Lines from Disk (A) commands to write and read successive blocks of text until the entire file has passed through the edit buffer.

The A command alone has no effect if the edit buffer is 75 percent or more full. The W command must be used to write lines to the output file and delete them from the buffer; then the A command can read new lines from the input file and append them to the end of the text remaining in the buffer.

The *n* parameter specifies the number of lines to be read from the file. If *n* is omitted or is too large, EDLIN reads only enough lines to fill the editing buffer to 75 percent of its capacity.

## Examples

To append 200 lines from the disk file to the edit buffer, type

```
*200A  <Enter>
```

To append as many lines from the file as possible (until the edit buffer is 75 percent full), type

```
*A  <Enter>
```

## Message

**End of input file**
The last section of the file being edited has been read into the edit buffer.

# EDLIN: C

2.0 and later

Copy Lines

## Purpose

Copies one or more lines from one location in the edit buffer to another.

## Syntax

[*first*],[*last*],*destination*[,*count*]C

where:

| | |
|---|---|
| *first* | is the number of the first line to be copied. |
| *last* | is the number of the last line to be copied. |
| *destination* | is the number of the line before which the copied lines are to appear. |
| *count* | is the number of times to execute the copy operation. |

## Description

The Copy Lines (C) command copies one or more text lines, inserting the copied lines at another location in the edit buffer. The original lines that were copied are unchanged. EDLIN then renumbers the edit buffer and makes the first copied line at the destination the new current line.

The *first* and *last* line-number parameters define the block of lines to be copied. (Note that the first line number must be less than or equal to the last line number.) Either or both of these numbers can be omitted (in which case the current line number is used), but the commas must still be entered as placeholders. The *destination* parameter specifies the line before which the copied lines are to be inserted; it is not optional and must not fall within the range of line numbers specified by *first* and *last*. One of the special symbols . (current line) or # (end of buffer) or an expression relative to the current line number (+*n* or −*n*) can be used instead of absolute line numbers.

To replicate the line or lines multiple times, the copy operation can be repeated automatically with the optional parameter *count*. The default value for *count* is one.

## Examples

If the current line is line 10, to copy lines 10 through 15 and place the copied lines before line 5, type

```
*10,15,5C  <Enter>
```

or

```
*,15,5C  <Enter>
```

or

```
*,+5,-5C  <Enter>
```

If the current line is line 10, to place three copies of lines 10 through 15 before line 1, type

```
*10,15,1,3C  <Enter>
```

or

```
*,15,1,3C  <Enter>
```

or

```
*,+5,1,3C  <Enter>
```

## Messages

**Entry error**
The command line contained an error such as a first line number that was greater than the last line number or a destination line number that fell within the range *first,last.*

**Insufficient memory**
The edit buffer does not have sufficient room for EDLIN to carry out the specified command.

**Must specify destination line number**
No destination line number was specified in the command line; therefore, no changes were made to the edit buffer.

# EDLIN: D

1.0 and later

Delete Lines

## Purpose

Deletes one or more lines from the edit buffer.

## Syntax

[*first*][,*last*]D

where:

| | |
|---|---|
| *first* | is the number of the first line to delete. |
| *last* | is the number of the last line to delete. |

## Description

The Delete Lines (D) command removes one or more text lines from the edit buffer. The line after the last line deleted becomes the new current line.

The *first* and *last* line-number parameters define the block of lines to be deleted. (Note that the first line number must be less than or equal to the last line number.) Either or both of these numbers can be omitted (in which case the current line number is used), but a leading comma is required as a placeholder if *first* is omitted when *last* is present. One of the special symbols . (current line) or # (end of buffer) or an expression relative to the current line number (+*n* or −*n*) can be used instead of absolute line numbers.

## Examples

If the current line is line 10, to delete the current line, type

```
*10D  <Enter>
```

or

```
*D  <Enter>
```

If the current line is line 10, to delete lines 10 through 15, type

```
*10,15D  <Enter>
```

or

```
*,15D  <Enter>
```

or

```
*,+5D  <Enter>
```

If the current line is line 10, to delete all lines from the current line to the end of the buffer, type

```
*10,#D  <Enter>
```

or

```
*,#D  <Enter>
```

## Message

**Entry error**
The command line contained an error such as a first line number that was greater than the last line number.

# EDLIN: E

1.0 and later

End Editing Session

## Purpose

Saves the edited file to disk and exits from EDLIN.

## Syntax

E

## Description

The End Editing Session (E) command writes the contents of the edit buffer to the current directory of the disk in the current drive. If a previously existing file was being edited and there is any text remaining in the original file that has not yet passed through the edit buffer, EDLIN copies this text to the output file. EDLIN gives the newly edited file the same name as the original file and renames the original (unchanged) file with the extension .BAK. Any previous file with the same name and the extension .BAK is lost. EDLIN then returns to MS-DOS.

If the disk does not have enough space to hold the edited file in addition to the original file, EDLIN writes as much of the edited file as possible into a file with the extension .$$$; the remainder of the edited text is lost. The name and contents of the original file are left unchanged.

## Example

To end an editing session, type

```
*E  <Enter>
```

## Messages

**Disk full. Edits lost.**
The disk does not contain enough free space for the edited file. A partial file may have been created with the extension .$$$.

**File Creation Error**
The .BAK file is marked read-only, the root directory is full or cannot contain any more files, or the filename is the same as a volume label or directory name.

**No room in directory for file**
The file could not be saved because its destination was the root directory and the root directory is full.

**Too many files open**
MS-DOS was unable to open the .BAK file due to a lack of available system file handles. Increase the value of the FILES command in the CONFIG.SYS file.

# EDLIN: I

1.0 and later

Insert Lines

## Purpose

Inserts new lines into the edit buffer.

## Syntax

[*destination*]I

where:

*destination* is the number of the line before which text is to be inserted.

## Description

The Insert Lines (I) command enables insert mode and allows new text to be placed between previously existing lines of text. When insert mode is terminated, the first line following the inserted lines becomes the new current line.

EDLIN places the new text before the line specified by the *destination* parameter. If *destination* is omitted, EDLIN assumes the current line; if *destination* is larger than the number of lines in the edit buffer, EDLIN simply appends the new text after the actual last line. One of the special symbols . (current line) or # (end of buffer) or an expression relative to the current line number (+*n* or −*n*) can be used instead of an absolute line number.

After an I command, EDLIN issues a prompt consisting of the line number for the inserted text followed by a colon and an asterisk and continues to issue such prompts each time the Enter key is pressed until the user terminates insert mode by pressing Ctrl-C or Ctrl-Break.

## Examples

If the current line is line 10, to insert text before line 7, type

```
*7I  <Enter>
```

or

```
*-3I  <Enter>
```

To insert lines at the beginning of the buffer, type

```
*1I  <Enter>
```

To insert lines at the end of the buffer, type

```
*#I  <Enter>
```

## Message

**Insufficient memory**
The edit buffer does not have sufficient room for EDLIN to complete the specified command.

# EDLIN: L

1.0 and later

List Lines

## Purpose

Displays one or more lines from the edit buffer.

## Syntax

[*first*][,*last*]L

where:

| | |
|---|---|
| *first* | is the number of the first line to be displayed. |
| *last* | is the number of the last line to be displayed. |

## Description

The List Lines (L) command displays text lines on standard output. If the current line lies within the range of lines listed, EDLIN displays an asterisk next to its number. The current line is not changed.

The *first* and *last* line-number parameters define the block of lines to be listed. (Note that the first line number must be less than or equal to the last line number.) Either or both of these numbers can be omitted, but a leading comma is required as a placeholder if *first* is omitted when *last* is present. One of the special symbols . (current line) and # (end of buffer) or an expression relative to the current line number (+*n* or −*n*) can be used instead of absolute line numbers.

If only the first line number is specified, EDLIN displays text in 23-line increments starting with that number. If only the last line number is specified, EDLIN displays text beginning 11 lines before the current line and continuing to the specified last line. If no line numbers are specified in the command, EDLIN lists the 23 lines centered around the current line; if the current line number is less than 13, EDLIN lists the first 23 lines in the buffer.

## Examples

To display lines 20 through 30, type

```
*20,30L  <Enter>
```

If the current line is 20, to display the 23 lines centered around the current line, type

```
*L  <Enter>
```

EDLIN displays lines 9 through 31.

## Message

**Entry error**
The command line contained an error such as a first line number that was greater than the last line number.

# EDLIN: M

2.0 and later

Move Lines

## Purpose

Moves lines from one place in the edit buffer to another.

## Syntax

[*first*],[*last*],*destination*M

where:

| | |
|---|---|
| *first* | is the number of the first line to be moved. |
| *last* | is the number of the last line to be moved. |
| *destination* | is the number of the line before which the moved lines are to be inserted. |

## Description

The Move Lines (M) command transfers one or more text lines from one location in the edit buffer to another. EDLIN then deletes the original lines and renumbers the edit buffer. The first moved line becomes the new current line.

The *first* and *last* line-number parameters define the block of lines to be moved. (Note that the first line number must be less than or equal to the last line number.) Either or both of these numbers can be omitted (in which case the current line number is used), but the commas must still be entered as placeholders. The *destination* parameter specifies the line before which the moved lines are to be inserted; it is not optional and must not fall within the range of line numbers specified by *first* and *last*. One of the special symbols . (current line) or # (end of buffer) or an expression relative to the current line number (+*n* or −*n*) can be used instead of absolute line numbers.

## Example

If the current line is line 10, to move lines 10 through 15 and place them before line 5, type

```
*10,15,5M  <Enter>
```

or

```
*,15,5M  <Enter>
```

or

```
*,+5,-5M  <Enter>
```

## Messages

**Entry error**
The command line contained an error such as a first line number that was greater than the last line number or a destination line number that fell within the range *first,last.*

**Must specify destination line number**
No destination line number was specified in the command line; therefore, no changes were made to the edit buffer.

# EDLIN: P

2.0 and later

Display in Pages

## Purpose

Displays lines for viewing in successive screenfuls (pages).

## Syntax

[*first*][,*last*]P

where:

| | |
|---|---|
| *first* | is the number of the first line to be displayed. |
| *last* | is the number of the last line to be displayed. |

## Description

The Display in Pages (P) command displays text lines on standard output one screenful at a time. Unlike the List Lines (L) command, which has no effect on the current line, P causes the last line displayed to become the new current line. Thus, although the edit buffer is not actually organized into pages, the user can employ repeated P commands to sequentially view successive groups of lines.

The *first* and *last* line-number parameters define the block of lines to be listed; the display starts with the line specified by *first*. (Note that the first line number must be less than or equal to the last line number.) Either or both of these numbers can be omitted, but a leading comma is required as a placeholder if *first* is omitted when *last* is present. If omitted, *first* defaults to the line after the current line and *last* defaults to the line 23 lines after the current line. One of the special symbols . (current line) or # (end of buffer) or an expression relative to the current line number (+*n* or −*n*) can be used instead of absolute line numbers.

## Examples

If the current line is 20, to view the next page of lines in the edit buffer, type

```
*P  <Enter>
```

EDLIN displays 23 lines, beginning with line 21, and changes the current line to line 43.

To view successive pages of 23 lines, repeatedly type

```
*P  <Enter>
```

## Message

**Entry error**
The command line contained an error such as a first line number that was greater than the last line number.

# EDLIN: Q

1.0 and later

Quit

## Purpose

Terminates the editing session without saving the revised file.

## Syntax

Q

## Description

The Quit (Q) command causes EDLIN to exit without saving any of the changes made to the edited file during the session. The original file's name and contents are left unchanged and no new file is created.

To reduce the danger of accidentally losing the contents of the edit buffer, EDLIN prompts the user for confirmation before carrying out the Q command.

## Example

To quit an editing session, type

```
*Q  <Enter>
```

EDLIN issues a prompt for confirmation and, if the response from the user is *Y*, exits to MS-DOS without saving any changes made to the file during the session.

## Message

**Abort edit (Y/N)?**
This prompt is displayed in response to the Q command. Respond with *Y* to exit to MS-DOS without saving changes made to the file; respond with *N* to continue the editing session.

# EDLIN: R

1.0 and later

Replace Text

## Purpose

Replaces one string in the edit buffer with another.

## Syntax

[*first*][,*last*][?]R[*string1*][^Z*string2*]

where:

| | |
|---|---|
| *first* | is the number of the first line to be searched. |
| *last* | is the number of the last line to be searched. |
| ? | causes the user to be prompted for confirmation before each replacement is made. |
| *string1* | is the sequence of characters to be searched for. |
| ^Z | is a Control-Z character. |
| *string2* | is the sequence of characters to be substituted for *string1*. |

***Note:*** The character limit for the Replace Text command is 127 characters, including both strings and all other parameters.

## Description

The Replace Text (R) command substitutes one character string for another within a specified range of lines. The last line in which a replacement occurs becomes the new current line.

The *first* and *last* line-number parameters define the range of lines to be searched for strings to replace. (Note that the first line number must be less than or equal to the last line number.) Either or both of these numbers can be omitted, but a leading comma is required as a placeholder if *first* is omitted when *last* is present. If omitted, *first* defaults to the line after the current line and *last* defaults to the last line in the buffer. One of the special symbols . (current line) or # (end of buffer) or an expression relative to the current line number (+*n* or −*n*) can be used instead of absolute line numbers.

If *string1* is omitted, EDLIN uses the *string1* from the preceding R command; if there was no preceding R command, EDLIN displays an error message. If *string2* is omitted, EDLIN deletes all occurrences of *string1*. *string1* must be separated from *string2* by a Control-Z (^Z) character. If *string1* is omitted, a Control-Z character must still be included to mark the beginning of *string2*, but if *string2* is omitted when *string1* is present, the Control-Z character has no effect and is therefore optional. (The Control-Z character is entered by pressing Ctrl-Z or the F6 key.)

If the ? option is not included in the command line, EDLIN displays each line that contains a match *after* the replacement is carried out. If the ? option is used, EDLIN displays each line containing a match as it is found and prompts the user for confirmation *before* the string is replaced.

The matching operation is case sensitive; EDLIN carries out the substitution only on sequences of characters that match *string1* exactly. Wildcards are not permitted.

## Examples

If the current line is line 10, to replace all occurrences of the string *logical* with the string *bitwise* in lines 11 through 20, type

```
*11,20Rlogical^Zbitwise  <Enter>
```

or

```
*,20Rlogical^Zbitwise  <Enter>
```

To cause EDLIN to prompt for confirmation before replacing each string, type

```
*11,20?Rlogical^Zbitwise  <Enter>
```

or

```
*,20?Rlogical^Zbitwise  <Enter>
```

To delete all occurrences of the string *00H* in line 20, type

```
*20,20R00H^Z  <Enter>
```

## Messages

**Entry error**
The command line contained an error such as a first line number that was greater than the last line number.

**Insufficient memory**
The edit buffer has insufficient room for EDLIN to carry out the specified Replace Text command.

**Line too long**
The replacement would cause the line being edited to expand beyond 253 characters.

**Not found**
No occurrence or further occurrences of the string to be replaced were found in the specified range of lines.

**O.K.?**
If the ? option is used in the command line, this prompt is displayed each time a matching string is found. Respond with *Y* or press the Enter key to replace the string and continue searching; press any other key to leave the string unchanged and continue searching.

# EDLIN: S

1.0 and later

Search for Text

## Purpose

Searches the edit buffer for a character string.

## Syntax

[*first*][,*last*][?]S[*string*]

where:

| | |
|---|---|
| *first* | is the number of the first line to be searched. |
| *last* | is the number of the last line to be searched. |
| ? | causes the user to be prompted for confirmation before the search is terminated. |
| *string* | is the sequence of characters to be searched for (maximum 126 characters). |

## Description

The Search for Text (S) command searches for a character string within a specified range of lines. When a match is found, EDLIN displays the line containing the match and that line becomes the new current line. If no lines containing the specified string are found, EDLIN displays the message *Not found* and the current line number remains unchanged.

The *first* and *last* line-number parameters define the block of lines to be searched for strings. (Note that the first line number must be less than or equal to the last line number.) Either or both of these numbers can be omitted, but a leading comma is required as a placeholder if *first* is omitted when *last* is present. If omitted, *first* defaults to the line after the current line and *last* defaults to the last line in the buffer. One of the special symbols . (current line) or # (end of buffer) or an expression relative to the current line number (+*n* or −*n*) can be used instead of absolute line numbers.

If *string* is omitted, EDLIN uses the *string* from the last S command or *string1* from the last Replace Text (R) command instead.

If the ? option is not included in the command line, EDLIN displays the first line that contains a match for *string*, makes this the new current line, and terminates the search. If the ? option is used, EDLIN displays each line containing a match for *string* as it is found, followed by an *O.K.?* prompt. If the user responds with *Y* or presses the Enter key, EDLIN terminates the search; if the user presses any other key, the search continues.

The matching operation is case sensitive; EDLIN reports only sequences of characters that match *string* exactly. Wildcards are not permitted.

## Examples

If the current line is line 10, to find the first occurrence of the string *xyz* in lines 11 through 20, type

```
*11,20Sxyz  <Enter>
```

**or**

```
*,20Sxyz  <Enter>
```

To find a particular occurrence of *proc* in the edit buffer, type

```
*1,#?Sproc  <Enter>
```

EDLIN displays the first line containing *proc* and prompts with

```
O.K.?
```

Type *Y* or press Enter to stop the search; press any other key to continue the search.

## Messages

**Entry error**
The command line contained an error such as a first line number that was greater than the last line number.

**Not found**
No match or no further matches for *string* were found in the specified range of lines.

**O.K.?**
If the ? option is used in the command line, this prompt is displayed each time a matching string is found. Respond with *Y* or press the Enter key to stop searching; press any other key to continue searching.

# EDLIN: T

2.0 and later

Transfer Another File

## Purpose

Merges the contents of another file with the file in the edit buffer.

## Syntax

[*destination*]T[*drive*:][*path*]*filename*

where:

| | |
|---|---|
| *destination* | is the number of the line before which the text from *filename* is to be inserted. |
| *path* | is the location of the file to be merged (versions 3.0 and later). |
| *filename* | is the name of the disk file from which text is to be merged. |

## Description

The Transfer Another File (T) command merges the contents of a text file with the current contents of the edit buffer and then renumbers the contents of the edit buffer. The first line of the merged text becomes the current line.

The *destination* parameter specifies the line before which the transferred lines are to be inserted. If omitted, *destination* defaults to the current line. One of the special symbols . (current line) or # (end of buffer) or an expression relative to the current line number (+*n* or −*n*) can be used instead of an absolute line number.

The *filename* parameter specifies the file from which text is to be merged and can include a drive and, in versions 3.0 and later, a path. If a drive or path is not specified, the file to be merged into the edit buffer with the T command *must* be in the current directory of the current drive.

## Example

If the current line is line 10, to merge the contents of the file named KEYDEFS.C before line 10 of the edit buffer, type

```
*10Tkeydefs.c  <Enter>
```

or

```
*Tkeydefs.c  <Enter>
```

## Messages

**File not found**
The specified filename does not exist in the current or specified location.

**Not enough room to merge the entire file**
The space available in the edit buffer is not sufficient to hold the entire file named in the T command. Use the Write Lines to Disk (W) command to partially empty the edit buffer.

# EDLIN: W

1.0 and later

Write Lines to Disk

## Purpose

Writes lines from the edit buffer to the disk.

## Syntax

[*n*]W

where:

*n* is the number of lines to be written to the file.

## Description

If the file being edited is too large to fit into the edit buffer, EDLIN ordinarily reads only enough text to fill 75 percent of the buffer when it opens the file, reserving 25 percent of the buffer for changes and additions to the text. The user must then employ the Write Lines to Disk (W) command and the Append Lines from Disk (A) command to transfer successive blocks of text from the disk until the entire file has passed through the edit buffer. The W command causes EDLIN to write lines to the disk file and delete them from the buffer; then the A command can read new lines from the input file, placing them after the end of the text remaining in the buffer.

The *n* parameter specifies the number of lines to be written to the output file; if *n* is omitted or is larger than the number of lines in the edit buffer, EDLIN writes only enough lines to leave the edit buffer about 25 percent full. EDLIN then renumbers the lines remaining in the edit buffer so that the first remaining line becomes line number one.

## Examples

To write 200 lines from the edit buffer to disk (effectively deleting those lines from the buffer), type

```
*200W  <Enter>
```

To write lines from the edit buffer to the disk until the edit buffer is only 25 percent full, type

```
*W  <Enter>
```

# EXIT

Terminate Command Processor

2.0 and later

Internal

## Purpose

Terminates a secondary copy of the command processor.

## Syntax

EXIT

## Description

Many communications programs, word processors, database managers, and other application programs load and execute a secondary copy of the system's command processor (COMMAND.COM) to let the user carry out MS-DOS commands without losing the context of the work in progress. Secondary copies of the command processor are also commonly used to execute one batch file under the control of another. (For more information about secondary copies of the command processor, *see* USER COMMANDS: COMMAND.)

The EXIT command cancels a secondary command processor. The terminating processor displays no message and control returns directly to the parent program or command processor.

EXIT has no effect on the currently executing command processor if it was loaded with the /P (permanent) switch or if it is the original command processor (the one loaded during system initialization, when the computer was turned on or restarted).

The EXIT command also allows the user to choose Close from the system menu if a COMMAND window is open under Microsoft Windows.

## Example

To terminate the currently executing command processor, type

```
C>EXIT  <Enter>
```

## Message

**Bad command or filename**
The EXIT command did not exist in versions earlier than 2.0, so MS-DOS attempted to execute a nonexistent program named EXIT instead.

# FC

2.0 and later

Compare Files

External

## Purpose

Compares two files and lists the differences on standard output.

## Syntax

FC [/A] [/C] [/L] [/LB*n*] [/N] [*/nnnn*] [/T] [/W] [*drive*:]*pathname1* [*drive*:]*pathname2*

or

FC [/B] [*drive*:]*pathname1* [*drive*:]*pathname2*

where:

| | |
|---|---|
| *pathname1* | is the name and location of the first file to be compared, optionally preceded by a drive; wildcard characters are not permitted. |
| *pathname2* | is the name and location of the second file to be compared, optionally preceded by a drive; wildcard characters are not permitted. |
| /A | causes FC to abbreviate the output when comparing ASCII text files (version 3.2). |
| /B | causes a byte-by-byte (binary) comparison; may not be used with any other switch (default when file extension is .EXE, .COM, .SYS, .OBJ, .LIB, or .BIN). |
| /C | causes FC to ignore case when comparing alphabetic characters. |
| /L | causes a line-by-line comparison of two ASCII text files (default when file extension is not .EXE, .COM, .SYS, .OBJ, .LIB, or .BIN) (version 3.2). |
| /LB*n* | sets the size of the internal line buffer to *n* lines (default = 100) (version 3.2). |
| /N | includes line numbers on the output of an ASCII file comparison (version 3.2). |
| */nnnn* | is the number of lines that must match to resynchronize during an ASCII file comparison (default = 2; in versions 2.0 through 3.1, range = 1–9, default = 3). |
| /T | causes FC to compare tabs in text files literally (default = tabs expanded to spaces, with stops at each eighth character position) (version 3.2). |
| /W | causes FC to ignore spaces, tabs, and blank lines in text files. |

## Description

The FC utility compares two text files containing lines of ASCII text delimited by new-line characters or two binary files containing data of any type (such as executable programs).

The differences between the two files are listed on standard output, which defaults to the video display but can be redirected to another character device or a file or can be piped to another program.

The FC program first examines the extensions of the two files being compared and, in most cases, selects the appropriate type of comparison automatically. However, the /B switch can be used to force a binary, or byte-by-byte, comparison of the two files named; the /L switch can be used to force a line-by-line comparison. When the /B switch is present, use of the /L, /N, and */nnnn* switches causes an error message to be displayed; any other switches in the command line are ignored.

When comparing ASCII text files, FC loads a buffer with sequential sets of lines from each file and compares the two sets. The size of this buffer defaults to 100 lines but can be modified by including the /LB*n* switch in the command line. If differences are found, the name of the first file, the last matched line, and any mismatched lines from that file are displayed, followed by the first rematched line; then the name of the second file, the last matched line, and any mismatched lines are displayed, followed by the first rematched line from that file. The number of consecutive matching lines that must be detected in order for FC to consider the files resynchronized is controlled with the */nnnn* switch; the default is 2.

If no lines match, if no lines match after the first mismatch, or if the number of mismatched lines exceeds the size of the line buffer, FC displays the message *Resynch failed. Files are too different* (or ****Files are different**** in versions 2.x and 3.0) and terminates.

The /C, /T, and /W switches modify the way in which two text files are compared. The /C switch causes FC to ignore case when comparing alphabetic characters. The /T switch causes FC to compare tab characters (ASCII code 09H) literally, rather than expand them to spaces before comparing corresponding lines. Finally, the /W, or whitespace, switch causes FC to ignore spaces, tabs, and blank lines during the comparison.

The /A and /N switches control the format of the listing of differences between the two text files. The /A switch causes FC to compress the listing of each mismatched set of lines to the first and last lines of each set, separated by ellipsis points. The /N switch causes FC to include the line numbers of the mismatched lines in the display.

During a binary comparison of two files, FC's buffer is reloaded as many times as is necessary to compare the complete files. Unlike the procedure with text-file comparisons, no attempt is made to resynchronize the data if a mismatch is detected and, regardless of the number of mismatches, the comparison process is not terminated. Any differences are displayed with the offset from the start of the file and the actual data from each file. If one file is shorter than the other, FC also displays a warning message at the end of the comparison.

The FC command is present only in MS-DOS. PC-DOS versions 1.0 and later provide a similar function in the COMP command.

## Examples

Assume that FILE1.TXT and FILE2.TXT are in the current directory on the disk in the current drive and that they contain the following lines:

| FILE1.TXT | FILE2.TXT |
|---|---|
| First line. | First line. |
| Second line. | Second line. |
| Third line. | Third line. |
| Fourth line. | Fourth line. |
| Fifth line. | Sixth line. |
| Sixth line. | Fifth line. |
| Seventh line. | Seventh line. |
| Eighth line. | Eighth line. |
| Ninth line. | Ninth line. |
| Tenth line. | Tenth line. |

To compare these files line by line, type

```
C>FC FILE1.TXT FILE2.TXT  <Enter>
```

This will result in the following display:

```
***** file1.txt
Fourth line.
Fifth line.
Sixth line.
Seventh line.
***** file2.txt
Fourth line.
Sixth line.
Fifth line.
Seventh line.
*****
```

To compare the same two files and produce an abbreviated listing of differences that includes line numbers, type

```
C>FC /A /N FILE1.TXT FILE2.TXT  <Enter>
```

This will result in the following display:

```
***** file1.txt
    4:  Fourth line.
...
    7:  Seventh line.
***** file2.txt
    4:  Fourth line.
...
    7:  Seventh line.
*****
```

Assume that two binary files, FILE1.BIN and FILE2.BIN, are the same length and contain only the following three differences:

| Offset | FILE1.BIN | FILE2.BIN |
|---|---|---|
| 19H | 04H | 03H |
| 33H | 4AH | 4BH |
| 42H | 52H | 51H |

To compare these two binary files, type

```
C>FC /B FILE1.BIN FILE2.BIN  <Enter>
```

This will result in the following display:

```
00000019: 04 03
00000033: 4A 4B
00000042: 52 51
```

***Note:*** The use of the /B switch in this example is optional; binary comparison is the default when .BIN files are compared.

## Messages

***filename* longer than *filename***
After all the corresponding data in the two files was compared, data remained in one of the files.

**cannot open *filename* - No such file or directory**
The specified file cannot be found or does not exist.

**DOS 2.0 or later required**
FC does not work with versions of MS-DOS earlier than 2.0.

**Incompatible switches**
The /B switch was used in combination with one or more of the other switches.

**Incorrect DOS version**
The version of FC is not compatible with the version of MS-DOS that is running.

**no differences encountered**
The two files being compared are identical.

**out of memory**
The available memory in the transient program area is insufficient to compare the two files.

**Resynch failed. Files are too different**
The number of mismatched lines in an ASCII file comparison exceeded the number of lines that can be loaded into FC's comparison buffer (which by default is 100 lines). Rerun the comparison using the /LB*n* switch to allocate a larger buffer.

**usage: fc [/a] [/b] [/c] [/l] [/lbNN] [/w] [/t] [/n] [/NNNN] file1 file2**
The command line included an invalid switch or FC was entered without any switches or other parameters.

# FDISK

3.2

Configure Fixed Disk

External No Net

## Purpose

Configures an MS-DOS partition on a fixed disk. This command is included with PC-DOS beginning with version 2.0.

## Syntax

FDISK

## Description

A fixed disk can be divided into areas of contiguous tracks, or partitions, that are used by different operating systems. A master control record (partition table) on the disk specifies the ID number and the starting and ending disk tracks for each partition. Each fixed disk can have as many as four partitions, but only one partition can be active (bootable) at any given time.

The FDISK utility is a menu-driven program that adds or deletes an MS-DOS partition on a fixed disk, selects one partition as active, and displays the size and status of all partitions. With most implementations of MS-DOS, each fixed disk can contain only *one* MS-DOS partition.

After an MS-DOS partition is created, the FORMAT command must be used to initialize the partition's directory structure. To make it possible to start the computer from the MS-DOS partition on the fixed-disk drive, the /S switch must be used with FORMAT to transfer the operating-system files and the MS-DOS partition must be the active partition.

***Warning:*** If the MS-DOS partition is deleted, any files stored in the partition are irretrievably lost.

## Examples

To display the current partitioning of the fixed disk, type

```
C>FDISK  <Enter>
```

The FDISK utility then displays the following menu:

```
Fixed Disk Setup Program Version 0.02
(C) Copyright Microsoft, 1985.

FDISK Options

Choose one of the following:

    1.  Create DOS Partition
    2.  Change Active Partition

    3.  Delete DOS Partition
    4.  Display Partition Data

Enter choice:[1]

Press ESC to return to DOS
```

***Note:*** A fifth option, *Select Next Fixed Drive*, will appear if more than one fixed disk is installed in the system.

Choose option 4 (*Display Partition Data*). FDISK then displays the partition data for the disk in the following form:

```
Display Partition Information

Partition Status  Type   Start  End  Size
    1       A      DOS       0  613   614
Total disk space is 614 cylinders.

Press ESC to return to FDISK Options
```

Assume that the low-level (hardware) formatting for fixed-disk drive C has just been completed by using the drive manufacturer's setup utility. To establish a bootable MS-DOS partition on the disk, type

```
A>FDISK  <Enter>
```

When the menu is displayed, press Enter to choose option 1 (*Create DOS Partition*). FDISK responds with the following message:

```
Create DOS Partition

Do you wish to use the entire fixed
disk for DOS (Y/N).................?[Y]

Press ESC to return to FDISK Options
```

To partition the entire fixed disk for MS-DOS, press Enter to select *Y* (the default). When the FDISK main menu is again displayed, choose option 4 (*Display Partition Data*) to verify that the MS-DOS partition has in fact been established on the fixed disk.

## Messages

***n* is not a choice. Please enter Y or N.**
The response to an FDISK prompt requiring a yes or no answer was not *Y* or *N*.

***n* is not a choice. Please enter a choice**
The response to an FDISK prompt requiring a number was not in the proper range or was not a number.

**DOS partition created**
A new MS-DOS partition has been established on the fixed disk. Use the FORMAT utility to create a directory structure in that partition.

**DOS partition deleted**
The previously existing MS-DOS partition on the fixed disk has been deleted. Any files contained in the partition are irretrievably lost.

**DOS 2.0 or later required**
FDISK does not work with versions of MS-DOS earlier than 2.0.

**Do you wish to use the entire fixed disk for DOS (Y/N).....?[Y]**
Option 1, *Create DOS Partition*, has been chosen from the main menu. Respond with *Y* or press Enter to use all available cylinders for a single DOS partition; respond with *N* to specify that only part of the fixed disk should be used.

**Enter starting cylinder number..:[*n*]**
Option 1, *Create DOS Partition*, has been chosen from the main menu and the user has responded *N* to the *Do you wish to use the entire fixed disk for DOS?* prompt. This message then prompts for the starting cylinder number of the DOS partition being created.

**Enter the number of the partition you want to make active.....:[*n*]**
Option 2, *Change Active Partition*, has been chosen from the main menu and this message prompts the user to enter the number of the partition that will become the active partition.

**Error loading operating system**
An error occurred while attempting to start the system from the fixed disk. Attempt to restart the system. If that fails, start the system from a floppy disk and use the SYS command to copy a new set of the operating-system files to the fixed disk.

**Error reading fixed disk**
An unrecoverable hardware error was encountered while FDISK was reading data from the fixed disk. The disk may require a low-level (hardware) formatting operation before FDISK can be used; this is usually performed with a special utility program provided by the drive manufacturer.

**Error writing fixed disk**
An unrecoverable hardware error was encountered while FDISK was writing the new partition control record to the fixed disk. Test the fixed disk with hardware diagnostics before further use.

**Fixed disk already has a DOS partition.**
The specified fixed disk already contains an MS-DOS partition. Be sure that the correct fixed disk has been selected before proceeding.

**Incorrect DOS version**
The version of FDISK is not compatible with the version of MS-DOS that is running.

**Invalid partition table**
The fixed disk's partition table is invalid and the operating system could not be loaded from the fixed disk during system initialization. Restart the computer using a floppy disk and rerun FDISK to determine and correct the problem.

**Missing operating system**
The DOS partition is the active partition, but it does not contain the operating system. (This message occurs only during system startup.) Use the SYS command to install the operating system.

**No DOS partition to delete.**
The fixed disk does not contain an MS-DOS partition.

**No fixed disks present**
FDISK cannot detect a fixed disk in the system. This may reflect a hardware problem with the fixed disk or its controller.

**No partitions defined.**
This informational message is displayed after the user has chosen option 4, *Display Partition Data*, to indicate that no partitions are currently defined.

**No partitions to make active**
The fixed disk has not been previously partitioned using FDISK; therefore, an active partition cannot be selected.

**No space for a *nnn* cylinder partition.**
The fixed disk does not have enough free cylinders to create the desired partition.

**No space to create a DOS partition.**
The fixed disk does not have enough free cylinders to create an MS-DOS partition.

**Partition *n* is already active**
The selected partition is already active (bootable); therefore, no action was taken.

**Partition *n* made active**
This informational message indicates that the selected partition has been made the active partition.

**System will now restart**
**Insert DOS diskette in drive A:**
**Press any key when ready...**
The DOS partition has successfully been created. Strike any key and the system will restart from the disk in drive A.

**The current active partition is *n*.**
This informational message indicates which partition is currently bootable.

**The table partition can't be made active.**
The master partition record cannot be made bootable.

**Total disk space is *nnn* cylinders.**
This informational message indicates the total number of cylinders on the fixed disk.

**Total disk space is *nnn* cylinders.**
**Maximum available space is *nnn***
**cylinders at *n*.**
The user has responded *N* to the *Do you wish to use the entire fixed disk for DOS?* prompt and this informational message indicates how much space is available for the DOS partition.

**Warning: Data in the DOS partition**
**will be lost. Do you wish to**
**continue.....?[N]**
If the MS-DOS partition is deleted, all files within the partition are lost. Be sure that the files are backed up to another disk before proceeding. Respond with *N* to return to the FDISK main menu; respond with *Y* to delete the DOS partition and lose any files within it.

# FIND

Find Character String

2.0 and later

External

## Purpose

Searches the character stream from a file or from standard input for a string and displays any lines that contain the string on standard output.

## Syntax

FIND [/C] [/N] [/V] *"string"* [[*drive*:][*path*]*filename*] [[*drive*:][*path*]*filename* ...]

where:

| | |
|---|---|
| *string* | is the character string to be searched for, always enclosed in quotation marks; case is significant. |
| *filename* | is the name of the file to be searched, optionally preceded by a drive and/or path; wildcard characters are not permitted. |
| /C | displays only the count of the lines containing *string*. |
| /N | includes the relative line number with each line. |
| /V | displays only those lines that do *not* contain *string*. |

## Description

The FIND command searches for all occurrences of a specified string in one or more files (or from standard input). Normally, FIND copies each line in which the string is found to standard output, which defaults to the video display but can be redirected to a file or another character device or can be piped to another program.

The string to be searched for must be enclosed in quotation marks. If the search string itself contains sets of quotation marks, each of those sets of quotation marks must be surrounded by an additional set of quotation marks. FIND's string search is case sensitive.

The search string can be followed by the names of one or more source files; these filenames cannot include wildcards. If no filename is supplied, FIND reads lines from standard input; unless input has been redirected from a file or from the output of another program, this means that FIND reads input from the keyboard. (Keyboard input is terminated by pressing Ctrl-Z or F6 followed by Enter.)

The /C switch counts the total number of lines in which the string appears and sends the count, rather than the lines themselves, to standard output. If the /C switch is used with /V, only the total count of lines that do *not* contain the specified search string is displayed. If both /C and /N are included in the same FIND command, the /N is ignored.

The /N switch includes a relative line number with each line sent to standard output. This is especially helpful when the output of FIND is to be used as a guide to editing the files.

The /V switch reverses the action of FIND so that it copies to standard output all lines that do *not* include the specified string.

## Examples

To find and display all lines in the files BREAK.ASM, TALK.ASM, and SHELL.ASM that contain the string *es:*, type

```
C>FIND "es:" BREAK.ASM TALK.ASM SHELL.ASM  <Enter>
```

To find and display all lines in the file STORY.TXT that contain the string *he said "no"*, type

```
C>FIND "he said ""no""" STORY.TXT  <Enter>
```

To search the file \SOURCE\MENUMGR.ASM on the current drive and display all lines that do not contain the string *Error*, type

```
C>FIND /V "Error" \SOURCE\MENUMGR.ASM  <Enter>
```

To obtain a listing on the printer of the lines in the file SHELL.ASM in the current directory of the current drive that contain the string *proc*, including line numbers, type

```
C>FIND /N "proc" SHELL.ASM > PRN  <Enter>
```

To search for all lines that contain two strings, pipe the output of one FIND command to be the input of another. For example, to find only those lines in the file MENUMGR.ASM in the current directory of the current drive that contain both the strings *MOV* and *AX*, type

```
C>FIND "MOV" MENUMGR.ASM | FIND "AX"  <Enter>
```

## Messages

**---------- *filename***
This informational message gives the name of the file that is currently being searched.

**FIND: Access denied**
The specified file is locked or being accessed by another application.

**FIND: File not found *filename***
The specified file does not exist or the path or drive is not correct.

**FIND: Invalid number of parameters**
The command line did not include a search string.

**FIND: Invalid Parameter *option***
The command line included an invalid switch.

**FIND: Read error in *filename***
A disk error occurred during processing of the specified file.

**FIND: Syntax error**
The command line included an invalid search string. The string must be enclosed in quotation marks.

**Incorrect DOS version**
The version of FIND is not compatible with the version of MS-DOS that is running.

# FORMAT

Initialize Disk

1.0 and later

External No Net

## Purpose

Prepares a disk for use by initializing the directory and file allocation table (FAT).

## Syntax

FORMAT [*drive*:] [/S] (versions 1.x)

or

FORMAT [*drive*:] [/O] [/V] [/S] (versions 2.0–3.1)

or

FORMAT *drive*: [/1] [/4] [/8] [/N:*n*] [/T:*n*] [/V] [/S] (version 3.2)

or

FORMAT *drive*: [/1] [/B] [/N:*n*] [/T:*n*] (version 3.2)

where:

| | |
|---|---|
| *drive* | is the location of the disk to be formatted. |
| /1 | formats a single-sided disk in a double-sided disk drive. |
| /4 | formats a standard double-sided, double-density disk (360 KB) on a quad-density disk drive. |
| /8 | formats a disk with 8 sectors per track. |
| /B | formats a disk with 8 sectors per track and preallocates space for the hidden operating-system files. |
| /N:*n* | formats a disk with *n* sectors per track. |
| /O | formats a disk that is compatible with PC-DOS versions 1.x. |
| /S | creates a system (bootable) disk; for most implementations of FORMAT, this must be the last switch in the command line. |
| /T:*n* | formats a disk with *n* tracks. |
| /V | allows a volume label to be assigned to the disk after formatting. |

***Note:*** Each OEM determines which switches will be supported by the FORMAT utility included with the versions of MS-DOS sold with its computers.

## Description

The FORMAT command effectively erases any existing data on a disk and creates a new root directory and file allocation table. Each sector of the disk is checked for defects and unusable sectors are marked so that they will not be assigned to files.

If the *drive* parameter is not supplied, the current or default drive is formatted. (A drive letter *must* be specified with version 3.2.) With versions 3.0 and later, the FORMAT program displays a warning if the drive to be formatted is a fixed disk and asks for confirmation before continuing.

When the formatting operation is complete, FORMAT displays the total amount of disk space, the number of bytes lost to defective sectors, the space reserved for or occupied by the hidden operating-system files (if the /B or /S switch was used), and the remaining free disk space. If a floppy disk was formatted, FORMAT then prompts the user to select between formatting another disk and returning to MS-DOS.

Normally, the type of disk drive determines the format that is given to a disk. For example, if a disk is formatted in a standard double-sided, double-density drive, the format defaults to double-sided, 40 tracks per side, 9 sectors per track. The version-specific default formats are 9 or 15 sectors per track with versions 3.0 and later, depending on the drive type; 9 sectors per track with versions 2.x; and 8 sectors per track with versions 1.x. The /1, /4, /8, /N:*n*, and /T:*n* switches can be used to override the default format in some cases. (Not all combinations of /N:*n* and /T:*n* are supported on all hardware.)

***Note:*** A disk formatted with the /4 switch might not be reliably read on a single- or double-sided double-density drive.

The /S switch creates a system (bootable) disk that contains a copy of the operating system. After the format operation is complete, the two hidden files IO.SYS and MSDOS.SYS (or IBMBIO.COM and IBMDOS.COM in PC-DOS) and the nonhidden file COMMAND.COM are copied to the newly formatted disk. Most implementations of FORMAT require that the /S switch, if used, be the last switch in the command line.

The /V switch allows a volume label to be assigned to the new disk. After formatting is complete, FORMAT prompts the user for a volume name, which can be as many as 11 characters. (The characters * ? / \ . , ; : + = < > [ ] and tab are not permitted in a volume label.) Volume labels are displayed by the DIR, CHKDSK, TREE, and VOL commands and, with MS-DOS versions 3.1 and later and PC-DOS versions 3.0 and later, can be modified with the LABEL command after the disk has been formatted.

The /O switch causes FORMAT to write an 0E5H byte at the start of each directory entry so that the resulting disk is compatible with MS-DOS and PC-DOS versions 1.x.

The /B switch formats a disk for 8 sectors per track and reserves room on the disk for the operating-system files. The operating system can then be transferred to the disk with the SYS command to make the disk bootable. The /B switch cannot be used in the same FORMAT command line as the /V or /S switch.

***Warning:*** Disks in drives affected by an ASSIGN, JOIN, or SUBST command should not be formatted. Disks cannot be formatted over a network.

## Return Codes

0 The FORMAT operation was successful.
3 The program was terminated by entry of a Ctrl-C or Ctrl-Break.
4 The program was terminated because of a fatal system error (any error other than 0, 3, or 5).
5 The program was terminated by an *N* response to the fixed-disk prompt *Proceed with FORMAT (Y/N)?*

***Note:*** Return codes are available with MS-DOS version 3.2.

## Examples

To format the disk in drive B, type

```
C>FORMAT B:  <Enter>
```

In response, FORMAT displays the following message:

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

With versions earlier than 3.2, FORMAT then displays the message

```
Formatting ...
```

after the Enter key is pressed, to show that the formatting operation is in progress. With version 3.2, FORMAT displays the message

```
Head: n Cylinder: nn
```

instead, to show the progress of the formatting operation. With all versions, FORMAT displays the following messages if the formatting operation is successful:

```
Format complete
    362496 bytes total disk space
    362496 bytes available on disk

Format another (Y/N)?
```

The byte values may vary depending on the drive type or the switches used in the command line. If bad sectors were encountered during the format operation, FORMAT also displays the number of bytes in bad sectors.

***Note:*** The *Format complete* message overwrites the head/cylinder status line but is appended to the *Formatting ...* status line.

To format and assign a volume label to the disk in drive B, type

```
C>FORMAT B: /V  <Enter>
```

After the usual formatting messages, FORMAT prompts as follows:

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

The user can then enter a volume name of as many as 11 characters (except * ? / ¦ . , ; : + = < > [ ] or tab), followed by a press of the Enter key.

To format the disk in drive B and make it a system (bootable) disk, type

```
C>FORMAT B: /S  <Enter>
```

FORMAT initializes the disk in the usual manner and then copies the two files containing the operating system (IO.SYS and MSDOS.SYS or IBMBIO.COM and IBMDOS.COM) and the file COMMAND.COM onto the disk. When the formatting operation is completed on a 360 KB floppy disk, the following messages appear:

```
Format complete
System transferred

    362496 bytes total disk space
     62464 bytes used by system
    300032 bytes available on disk

Format another (Y/N)?
```

The number of bytes used by the system will vary with the version of MS-DOS in use.

## Messages

***n* bytes total disk space**
***n* bytes used by system**
***n* bytes in bad sectors**
***n* bytes available on disk**
When formatting is complete, FORMAT displays this message with information about space available on the disk. The *bytes used by system* line will not appear if the /S switch was not specified; the *bytes in bad sectors* line will not appear if no bad sectors were found.

**Attempted write-protect violation**
The disk to be formatted is write protected. Remove the write-protect tab and respond with a *Y* to the *Format another (Y/N)?* prompt.

**Cannot find System Files**
The /S switch was used and FORMAT was unable to find the necessary system files in the default drive or in drive A.

**Cannot FORMAT a Network drive**
An attempt was made to format a disk in a drive that has been assigned to a network.

**Cannot format an ASSIGNed or SUBSTed drive.**
An attempt was made to format a disk in a drive affected by an ASSIGN or SUBST command.

**Disk unsuitable for system disk**
Defective sectors were detected on the tracks where the operating-system files would normally reside on a bootable disk. Such a disk should be used only for data files, if at all.

**Drive letter must be specified**
A drive letter must be specified when using version 3.2.

**Drive not ready**
The floppy-disk drive is empty or the drive door is not closed.

**Enter current Volume Label for drive *X*:**
The specified drive is a fixed disk, so FORMAT prompts the user to enter the current volume label for verification.

**Error in IOCTL call**
An internal system error occurred when a pre-version-3.2 block-device driver was used with version 3.2 of FORMAT.

**Error reading partition table**
FORMAT was unable to read the fixed disk's partition table. Use FDISK on the fixed disk and then try the FORMAT command again.

**Error writing directory**
FORMAT was unable to create a directory on the disk it is attempting to format. The disk is defective.

**Error writing FAT**
FORMAT was unable to create the FAT on the disk it is attempting to format. The disk is defective.

**Error writing partition table**
FORMAT was unable to write the fixed disk's partition table. Use FDISK on the fixed disk and then try the FORMAT command again.

**Format another (Y/N)?**
At the end of a successful formatting operation or after a nonfatal error, this prompt offers the user the opportunity to format another disk using the same switches specified in the original FORMAT command. Respond with *Y* to format another disk; respond with *N* to return to MS-DOS.

**Format complete**
The formatting operation has ended. This message contains a number of space characters after it and is printed over the top of the head/cylinder status message, effectively erasing it.

**Format failure**
The formatting operation was not successful. (This message is usually preceded by another message telling the user why the format failed.) This message contains a number of space characters after it and is printed over the top of the head/cylinder status message, effectively erasing it.

**Format not supported on drive *X*:**
Device parameters that the computer cannot support were specified in the FORMAT command line.

**Formatting...**
This informational message indicates that the FORMAT operation is in progress (versions 1.0 through 3.1).

**Head: *n* Cylinder: *nn***
This informational message indicates the progress of the FORMAT command during the formatting operation (version 3.2).

**Incorrect DOS version**
The version of FORMAT is not compatible with the version of MS-DOS that is running.

**Insert DOS disk in drive *X*:**
**and strike ENTER when ready**
The /S switch was specified in the FORMAT command line and the disk containing the FORMAT command does not also contain the hidden system files.

**Insert new diskette for drive *X*:**
**and strike ENTER when ready**
This prompt allows the user to change disks before the FORMAT operation continues.

**Insufficient memory for system transfer**
The command line included the /S switch, but available RAM is insufficient to hold the system files during the FORMAT operation.

**Invalid characters in volume label**
Certain characters (* ? / \ | . , ; : + = < > [ ] and tab) are not allowed in a volume name.

**Invalid device parameters from device driver**
The DEVICE or DRIVPARM device-driver parameters in the CONFIG.SYS file were incorrectly set or the fixed disk specified in the command line was formatted using MS-DOS versions 2.x without first running FDISK. FORMAT displays this message when the number of hidden sectors is not evenly divisible by the number of sectors per track (meaning that the partition does not start on a track boundary).

**Invalid drive specification**
The drive specified after the FORMAT command is not a valid drive.

**Invalid media or Track 0 bad - disk unusable**
One of the switches supplied in the command line is not valid for the drive containing the disk to be formatted (for example, the /8 switch for a quad-density floppy disk) or track 0 of the disk being formatted is unusable to the point that FORMAT is unable to create a directory or file allocation table (FAT).

**Invalid parameter**
One of the switches supplied in the command line is not valid or is not supported by the version of FORMAT being used.

**Invalid volume ID**
The volume label entered in response to the *Enter current Volume Label for drive X:* prompt was not the same as the current volume label. Use the VOL command to determine the current volume label.

**Non-System disk or disk error**
**Replace and strike any key when ready**
The command line contained a /S or /B switch, but the source disk does not contain the operating-system files.

**Not a block device**
The drive containing the disk to be formatted is not recognized by MS-DOS as a valid block device.

**Parameters not compatible**
Switches that cannot be used together were specified in the command line.

**Parameters not compatible with fixed disk**
One of the switches specified in the command line is not compatible with the specified drive.

**Parameters not supported**
One of the parameters specified in the command line is not supported by the version of FORMAT being used.

**Parameters not Supported by Drive**
The device driver for the specified drive does not support generic IOCTL function requests.

**Re-insert diskette for drive *X*:**
This message prompts the user to reinsert the disk being formatted into the specified drive.

**System transferred**
The system files IO.SYS and MSDOS.SYS (or IBMBIO.COM and IBMDOS.COM in PC-DOS) and the file COMMAND.COM have been successfully transferred to the newly formatted disk.

**Too many open files**
FORMAT was unable to write the volume label because insufficient system file handles were available. Increase the value of FILES in the CONFIG.SYS file.

**Volume label (11 characters, ENTER for none)?**
After formatting a disk with the /V option, FORMAT offers the user the opportunity to enter a volume label for the disk.

**Unable to write BOOT**
The first track of the disk or MS-DOS partition is bad and cannot be made bootable.

**WARNING, ALL DATA ON NON-REMOVABLE DISK**
**DRIVE *X*: WILL BE LOST!**
**Proceed with Format (Y/N)?**
If a fixed disk is specified as the disk to be formatted, FORMAT warns the user and gives the opportunity to cancel the FORMAT command (versions 3.0 and later).

# GRAFTABL

3.0 and later

Load Graphics Character Set

External

## Purpose

Installs a RAM-resident table of bitmaps that defines the screen appearance of character codes 128 through 255 in graphics mode.

## Syntax

GRAFTABL

## Description

On IBM PCs and compatibles in graphics display modes, the video-display BIOS routines (Interrupt 10H) display characters by writing bitmapped matrices of dots to the display. The dot pattern of each screen character's matrix is defined by an entry in a table of bitmaps. The table of bitmaps for the regular ASCII characters, coded 0 through 7FH (0–127), is permanently located in ROM and is always available for use by the system's video driver. The GRAFTABL utility contains a similar table of bitmaps for the upper (extended) characters, coded 80H through 0FFH (128–255). The GRAFTABL command loads this table into RAM and places the address of the table in the vector for Interrupt 1FH.

The GRAFTABL command is not needed for the IBM PCjr or for an enhanced graphics adapter; their ROM BIOS already contains tables of bitmaps for the extended character set.

GRAFTABL is a terminate-and-stay-resident (TSR) program; therefore, its installation reduces the amount of RAM available for use by application programs.

The GRAFTABL command can be executed only once after the computer has been turned on or restarted. An attempt to execute it again will result in an informational message stating that the graphics characters are already loaded.

## Example

To load the table of bitmaps for characters 80H through 0FFH (128–255) for use in graphics mode, type

```
C>GRAFTABL  <Enter>
```

## Messages

**DOS 2.0 or later required**
GRAFTABL does not work with versions of MS-DOS earlier than 2.0.

**Graphics characters already loaded**
The GRAFTABL command has already been executed since the system was turned on or restarted.

**Graphics characters loaded**
The table of bitmaps has been successfully loaded into RAM and the interrupt vector that points to the table has been initialized.

**Incorrect DOS version**
The version of GRAFTABL is not compatible with the version of MS-DOS that is running.

# GRAPHICS

3.2

Load Graphics Screen-Dump Program

External

## Purpose

Installs a resident program that can dump screen contents to the printer in graphics mode. This command is also available with PC-DOS versions 2.0 and later.

## Syntax

GRAPHICS (PC-DOS 2.x)

or

GRAPHICS [*printer*] [/B] [/R] (PC-DOS 3.0 and above)

or

GRAPHICS [*printer*] [/B] [/C] [/F] [/P *port*] [/R] (MS-DOS 3.2)

where:

*printer* is the type of printer to be supported, from the following list:

| | |
|---|---|
| COLOR1 | IBM Personal Computer Color Printer with black ribbon |
| COLOR4 | IBM Personal Computer Color Printer with red-green-blue-black (RGB) ribbon |
| COLOR8 | IBM Personal Computer Color Printer with cyan-magenta-yellow-black (CMY) ribbon |
| COMPACT | IBM Personal Computer Compact Printer |
| GRAPHICS | IBM Personal Computer Graphics Printer or compatible (the default) |

/B prints the background in color; valid only with the COLOR4 and COLOR8 printers.

/C centers the printout on the page.

/F flips (rotates) the printout 90 degrees.

/P *port* specifies which port the printer is attached to (1–3, where 1 = LPT1, 2 = LPT2, and 3 = LPT3).

/R prints the image as it appears on the screen (white characters on a black background) rather than reversed (the default, black characters on a white background).

## Description

The default system routine for dumping the screen to the printer (invoked by Shift-PrtSc) cannot interpret the display in graphics modes. The GRAPHICS command loads a more

sophisticated routine that can dump CGA-compatible graphics displays to several models of IBM graphics printers or compatibles.The GRAPHICS command is not compatible with the Hercules monochrome graphics card or with an enhanced graphics adapter in its enhanced display modes.

If the display is in 640 x 200 graphics mode, the screen dump is printed sideways (rotated 90 degrees). A 320 x 200 graphic can be rotated manually by specifying the /F switch in the command line; however, the image will be elongated horizontally. A rotated image is printed along the left side of the page, which is actually the top of the page in terms of image orientation. The /C option can be used to center a rotated 320 x 200 image on the page.

When used with a printer with a black ribbon, GRAPHICS produces screen dumps with as many as four shades of gray to represent the colors. When used with a printer with a color ribbon (type COLOR4 or COLOR8), GRAPHICS prints all the colors except the background color. With printer types COLOR4 and COLOR8, the /B switch can be used to print the background color also.

Ordinarily, the screen image being dumped is reversed from its appearance on the screen; that is, the light areas on the screen are dark on the printed output and vice versa. The /R switch produces a screen dump that is not reversed in this manner.

If the *printer* parameter is not included in the command line, the GRAPHICS program assumes an IBM Personal Computer Graphics Printer or compatible.

If two or more printers are attached to the system, the /P switch can be used to specify which printer GRAPHICS should use.

The GRAPHICS command is a terminate-and-stay-resident (TSR) program; therefore, its installation reduces the amount of RAM available for use by application programs.

## Examples

To load the graphics printing program for use with an IBM Personal Computer Graphics Printer or compatible connected to LPT2, type

```
C>GRAPHICS /P 2  <Enter>
```

***Note:*** A tab, a semicolon character (;), or an equal sign (=) can be used between the /P and the port number instead of a space.

To load the graphics printing program for use with the IBM Personal Computer Color Printer with an RGB ribbon and specify that the background color be printed, type

```
C>GRAPHICS COLOR4 /B  <Enter>
```

To load the graphics printing program for use with the IBM Personal Computer Compact Printer and specify that the images be printed sideways and centered on the page, type

```
C>GRAPHICS COMPACT /F /C  <Enter>
```

## Messages

**DOS 2.0 or later required**
GRAPHICS does not work with versions of MS-DOS earlier than 2.0.

**Incorrect DOS version**
The version of GRAPHICS is not compatible with the version of MS-DOS that is running.

**Unrecognized printer**
The printer type specified in the command line is invalid or the printer is not supported.

**Unrecognized printer port**
The port specified with the /P switch is not a number in the range 1 through 3 or an invalid separator character was used.

# JOIN

Join Disk to Directory

3.0 and later

External No Net

## Purpose

Joins the directory structure of a disk drive to a subdirectory on another drive.

## Syntax

JOIN [*drive1*: *drive2*:*path*]

or

JOIN *drive1*: /D

where:

| | |
|---|---|
| *drive1* | is the drive whose directory structure will be joined to a subdirectory of another drive. |
| *drive2*:*path* | is the drive and directory that will be used to reference files on *drive1*. |
| /D | cancels the effect of a previous JOIN command on *drive1*. |

## Description

The JOIN command allows the directory structure of a disk in one drive to be joined, or spliced, into an empty subdirectory of a disk in another drive. After a JOIN, the entire directory structure of the disk in *drive1*, starting at the root, together with all the files that it contains, appears to be the directory structure of the specified subdirectory on the disk in *drive2*; the drive letter for *drive1* is no longer available. If the directory at the end of the path on *drive2* already exists, it must not contain any files; if it does not exist, JOIN will attempt to create it.

The current directory status of *drive1* has no effect on the JOIN operation. Regardless of which directory or subdirectory is active when the JOIN command is entered, the entire directory structure, including the root directory, is joined to the subdirectory on the disk in *drive2*.

The /D switch cancels any previous JOIN command for a specific drive.

If the JOIN command is entered without parameters, it displays a list of all joins currently in effect.

***Warning:*** The JOIN command should not be used on drives affected by a SUBST or ASSIGN command. Similarly, the BACKUP, RESTORE, FORMAT, DISKCOPY, and DISKCOMP commands should not be used on drives affected by the JOIN command. Drives that have been redirected over a network cannot be joined.

## Examples

To join drive B to the subdirectory \DRIVEB on drive C, type

```
C>JOIN B: C:\DRIVEB  <Enter>
```

A subsequent JOIN command without parameters displays

```
B: => C:\DRIVEB
```

To then list the files in the root directory of the disk in drive B, type

```
C>DIR C:\DRIVEB  <Enter>
```

To cancel a previous JOIN command affecting drive B, type

```
C>JOIN B: /D  <Enter>
```

## Messages

**Cannot JOIN a network drive**
A drive assigned to a network cannot be joined to another drive.

**Directory not empty**
A drive cannot be joined to a directory that already contains files.

**DOS 2.0 or later required**
JOIN does not work with versions of MS-DOS earlier than 2.0.

**Incorrect DOS version**
The version of JOIN is not compatible with the version of MS-DOS that is running.

**Incorrect number of parameters**
There were missing, extra, or incorrect parameters in the command line.

**Invalid parameter**
A drive cannot be joined to the root directory of any drive.

**Not enough memory**
The available system memory is insufficient for MS-DOS to run the JOIN command.

# KEYB*xx*

Define Keyboard

3.2

External

## Purpose

Installs a table that defines the translation of keys to the extended character codes, replacing the default table in the ROM BIOS. This command is included with PC-DOS beginning with version 3.0.

## Syntax

KEYB*xx*

where:

*xx* is a code that selects a keyboard configuration:

| | |
|---|---|
| DV | Dvorak keyboard (MS-DOS only) |
| FR | French |
| GR | German |
| IT | Italian |
| SP | European Spanish |
| UK | United Kingdom English |

***Note:*** KEYB*xx* is hardware dependent; therefore, implementation of this command may vary for different OEM versions of MS-DOS.

## Description

The KEYB*xx* utility configures the keyboard for use with a language other than United States English, making available special characters that are appropriate for the specified country's language and currency. These special characters are represented by the extended character codes (128–255) that correspond to the characters implemented on the OEM's display adapter. (Both the KEYB*xx* and the GRAFTABL commands must be used to make these characters available in graphics modes on a color/graphics adapter.)

After KEYB*xx* is loaded, special accented characters not part of the language in use are also available through the use of dead keys — keys that are pressed and released before the letter key is pressed. The following dead keys are available on a United States English keyboard for an IBM PC, PC/XT, PC/AT, or strict compatible:

| Keyboard Program | Dead Key | Resulting Accent |
|---|---|---|
| KEYBGR (Germany) | + | ` |
| | = | ´ |
| KEYBFR (France) | [ | ^ |
| | { | ¨ |
| KEYBSP (Spain) | [ | ´ |
| | ] | ` |
| | { | ¨ |
| | } | ^ |
| KEYBUK (United Kingdom) | Not supported | |
| KEYBIT (Italy) | Not supported | |

The dead-key combinations supported are

| Keyboard Program | Combinations Supported |
|---|---|
| Germany | á é É í ó ú à è ì ò ù |
| France | ä Ä ë ï ö Ö ü Ü ÿ â ê î ô û |
| Spain | ä Ä ë ï ö Ö ü Ü ÿ á é É í ó ú à è ì ò ù â ê î ô û |
| United Kingdom | Dead key not supported |
| Italy | Dead key not supported |

On an IBM PC, PC/XT, PC/AT, or strict compatible, the key sequence Ctrl-Alt-F1 can be used at any time to return the keyboard to the default (United States English) configuration; the sequence Ctrl-Alt-F2 then returns the keyboard to the selected configuration.

KEYB*xx* should be loaded only once during an MS-DOS session; the computer should be restarted if KEYB*xx* is loaded for use with a different language.

KEYB*xx* is a terminate-and-stay-resident (TSR) utility and therefore reduces the amount of memory available to transient application programs (by approximately 2 KB). The only way to reclaim this memory is to restart the system.

## Example

To configure the keyboard for Germany, type

```
C>KEYBGR  <Enter>
```

## Messages

**Bad command or filename**
The selected keyboard does not exist or the program that configures the keyboard is not present on the disk.

**Incorrect DOS version**
The version of KEYB*xx* is not compatible with the version of MS-DOS that is running.

# LABEL

Modify Volume Label

3.1 and later

External No Net

## Purpose

Adds, alters, or deletes a volume label on a disk. This command is included with PC-DOS beginning with version 3.0.

## Syntax

LABEL [*drive*:][*label*]

where:

| | |
|---|---|
| *drive* | is any valid disk drive. |
| *label* | is a name up to 11 characters long. |

## Description

With MS-DOS versions 2.0 and later, each disk can have a name called a volume label, which is implemented as a special type of entry in the disk's root directory. With MS-DOS versions 2.x, this volume label can be assigned to a disk only at the time the disk is formatted, using the FORMAT command's /V switch. However, with PC-DOS versions 3.0 and later and MS-DOS versions 3.1 and later, the volume label can be added, modified, or deleted at any time using the LABEL command. (A disk's volume label can be displayed with the VOL command; the label is also included as part of the output from the CHKDSK, DIR, and TREE commands.)

If a new volume name is included in the LABEL command line, the disk's label is changed immediately. If LABEL is entered alone or with only a drive letter, a message is displayed giving the current volume label of the disk in the specified drive (or the default drive, if no drive letter is given) and prompting the user for a new label. (A volume label can be from 1 to 11 characters; it cannot contain any of the characters * ? / \ | . , ; : + = < > [ ] or tab.) If no new volume name is supplied (the user did not type a volume label before pressing Enter), LABEL prompts the user to indicate whether the previous volume label should be deleted. Existing files on the disk are in no way affected by the LABEL command.

The LABEL command cannot be used on a network drive. With MS-DOS version 3.2, the LABEL command also cannot be used on a disk in a drive that is affected by an ASSIGN or SUBST command.

## Examples

To give the volume label PAYROLL to the disk in drive B, type

```
C>LABEL B:PAYROLL  <Enter>
```

Note that LABEL immediately overwrites any existing volume label on drive B with the new name; no warning of an existing volume label is given.

To remove the volume label LEDGER from the disk in drive A, type

```
C>LABEL A:  <Enter>
```

The LABEL command displays

```
Volume in drive A is LEDGER
Volume label (11 characters, ENTER for none)?
```

Press the Enter key to receive the additional prompt

```
Delete current volume label (Y/N)?
```

Then respond with *Y* and Enter to remove the volume label from the disk in drive A.

## Messages

**Cannot LABEL a Network drive**
The disk drive specified in the command line cannot be a network drive.

**Cannot LABEL a SUBSTed or ASSIGNed drive**
The disk drive specified in the command line is currently affected by a SUBST or ASSIGN command (MS-DOS version 3.2).

**Delete current volume label (Y/N)?**
No volume label was entered in response to the volume-label prompt and a volume label already exists on the disk. Respond with *Y* to delete the current label; respond with *N* to terminate the command.

**Incorrect DOS version**
The version of LABEL is not compatible with the version of MS-DOS that is running.

**Invalid characters in volume label**
The characters * ? / \ | . , ; : + = < > [ ] and tab cannot be part of a volume label.

**Invalid drive specification**
The drive specified in the command line is not valid or does not exist in the system.

**No room in root directory**
The root directory of the disk in the designated drive is full and a volume label cannot be added. Delete a file or subdirectory from the root directory to make room for the label.

**Too many files open**
LABEL was unable to write the volume label because no system file handles were available. Increase the value of FILES in the CONFIG.SYS file.

**Volume in drive *X* has no label**
**Volume label (11 characters, ENTER for none)?**

or

**Volume in drive *X* is *xxxxxxxxxxx***
**Volume label (11 characters, ENTER for none)?**
This informational message informs the user of the current volume label and prompts the user to add, change, or delete it.

# MKDIR or MD

Make Directory

2.0 and later

Internal

## Purpose

Creates a new directory.

## Syntax

MKDIR [*drive*:][*path*]*new_directory*

or

MD [*drive*:][*path*]*new_directory*

where:

*new_directory* is a valid directory name, optionally preceded by an existing path and/or a disk drive.

## Description

The MKDIR command creates a directory, adding a branch to the hierarchical directory structure of the disk. If the name of the new directory is preceded by a path, indicating that the new directory is to be a subdirectory of that path, the specified path must already exist.

If *new_directory* is not preceded by an existing path or a backslash character (\), it is presumed to be relative to the current directory. If *new_directory* is preceded by a backslash alone, the directory created will be a subdirectory of the root directory, regardless of the current directory. The length of the full path (including *new_directory*) must not exceed 63 characters.

***Warning:*** The MKDIR command should not be used to create new directories on drives affected by an ASSIGN, JOIN, or SUBST command.

## Examples

To create a directory named SOURCE in the current directory of the disk in the current drive, type

```
C>MKDIR SOURCE  <Enter>
```

or

```
C>MD SOURCE  <Enter>
```

To create a directory named LETTERS in the existing directory named WORD (which is a subdirectory of the root directory) on the disk in drive D, type

```
C>MKDIR D:\WORD\LETTERS  <Enter>
```

or

```
C>MD D:\WORD\LETTERS  <Enter>
```

## Messages

**Invalid drive specification**
The drive specified in the command line is not valid or does not exist in the system.

**Invalid number of parameters**
The name of the new directory was not included in the MKDIR command line.

**Unable to create directory**
The specified directory cannot be created. This may be caused by a full disk (if the new directory would cause the current directory to be extended), a full root directory (if the new directory's parent is the root directory), the existence of a file or directory with the same name, or an invalid *new_directory* name.

# MODE

Configure Device

3.2

External

## Purpose

The MODE command has four distinct uses:

- To reconfigure a printer attached to a parallel port (LPT1, LPT2, or LPT3) for printing at 80 or 132 characters per line, 6 or 8 lines per inch, or both (if the printer supports these features). In this form, MODE can also be used to select a parallel printer other than the one attached to LPT1 for use as the default printer.
- To select another display or reconfigure the current display. Reconfiguration includes changing between 40-column and 80-column display, changing between monochrome and color display, centering the display on the screen, or any combination of these.
- To configure the baud rate, parity, and number of databits and stop bits of a serial communications port (COM1 or COM2) for use with a specific printer, modem, or other serial device.
- To redirect printer output from a parallel port to one of the serial ports, so that the serial port becomes the system's default printer port.

Because the syntax for each of these uses of MODE is different, they are discussed separately on the following pages.

Although each form of the MODE command can be issued at the system prompt, MODE commands are commonly used within the AUTOEXEC.BAT file to automatically perform any necessary reconfiguration each time the system is turned on or restarted.

The MODE command is included with PC-DOS beginning with version 1.0.

## Message

**Incorrect Version of MODE**
The version of MODE is not compatible with the version of MS-DOS that is running.

# MODE

3.2

Configure Printer

External

## Purpose

Sets characteristics for IBM-compatible printers connected to a parallel printer port (LPT1, LPT2, or LPT3). This form of the MODE command is included with PC-DOS beginning with version 1.0.

## Syntax

MODE LPT*n*[:][*cpl*] [,[*lpi*][,P]]

where:

| | |
|---|---|
| LPT*n* | is the parallel printer port (*n* = 1, 2, or 3). |
| *cpl* | is the number of characters per line (80 or 132, default = 80). |
| *lpi* | is the number of lines per inch (6 or 8, default = 6). |
| P | causes continuous retries when the printer is not ready. |

## Description

This form of the MODE command configures an IBM or compatible printer connected to parallel port *n*. Its effect on other printer types may vary. The command has the side effect of canceling any redirection that was previously applied to the specified port with a Redirect Printing MODE command.

The first parameter, LPT*n*, designates the parallel printer port to be configured (LPT1, LPT2, or LPT3). All the other parameters are optional.

The *cpl* parameter selects between printing 80 characters on a line (the default) and 132 characters on a line. The *lpi* parameter selects between 6 lines per inch (the default) and 8 lines per inch. (Note that the attached printer must be capable of printing 132 characters per line or 8 lines per inch and of understanding IBM-compatible printer-control codes; otherwise, specifying these values will have no effect.)

The last parameter in the command line, P, configures the system to retry output continuously (or until Ctrl-Break is pressed) if the printer is not ready or not on line (interpreted by the computer as a time-out error), rather than display an error message. (Note that if P is used and *lpi* is omitted, the comma preceding *lpi* must be specified.) Use of the P option causes part of the MODE program to become permanently resident in memory. (This option is not available in PC-DOS version 1.0.)

## Examples

To configure the printer on the first parallel port to print 132 characters per line, with 8 lines per inch, type

```
C>MODE LPT1:132,8  <Enter>
```

To configure the system to continually send output to the printer on the second parallel port if a time-out error occurs but to leave the other values at their defaults, type

```
C>MODE LPT2:,,P  <Enter>
```

## Messages

**DOS 2.0 or later required**
MODE does not work with versions of MS-DOS earlier than 2.0.

**Incorrect DOS version**
The version of MODE is not compatible with the version of MS-DOS that is running.

**Infinite retry of parallel printer timeout**
The P option was included in the command line and the system will continuously retry to send output to the printer attached to the specified port if it is not ready or not on line.

**INTERNAL ERROR in MODE application**
An internal error occurred in the MODE utility and the requested reconfiguration was not carried out.

**Invalid parameters**
The command line included an incorrect parallel-port specification or one of the configuration parameters was not correct.

**LPT*n*: set for 80**
The specified printer has been configured for 80 characters per line.

**LPT*n*: set for 132**
The specified printer has been configured for 132 characters per line.

**Printer error**
The configuration command could not be carried out because the printer is turned off, not ready, or not on line.

**Printer lines per inch set**
The printer has successfully been configured for the specified 6 or 8 lines per inch.

**Resident portion of MODE loaded**
The P option was specified in the command line and part of the MODE command has become permanently resident in memory, decreasing slightly the amount of memory available to other programs.

# MODE

3.2

Set Display Mode

External

## Purpose

Selects the active video adapter and its display mode or reconfigures the current display. This form of the MODE command is included with PC-DOS beginning with version 2.0.

## Syntax

MODE *display*

or

MODE [*display*],*shift*[,T]

where:

*display* is a video adapter and display mode from the following list:

| | |
|---|---|
| 40 | Color/graphics adapter, 40 characters per line |
| 80 | Color/graphics adapter, 80 characters per line |
| BW40 | Color/graphics adapter, 40 characters per line, color disabled from composite output |
| BW80 | Color/graphics adapter, 80 characters per line, color disabled from composite output |
| CO40 | Color/graphics adapter, 40 characters per line, color enabled |
| CO80 | Color/graphics adapter, 80 characters per line, color enabled |
| MONO | Monochrome adapter |

*shift* is R or L, to shift the display left or right one (40-column display) or two (80-column display) character positions.

T causes a test pattern to be displayed for screen alignment.

## Description

This form of the MODE command has two uses. The first is to select the active video adapter and its display mode (if more than one adapter is present in the system) or to reconfigure the current adapter. The second is to shift the screen display to the left or right to center it. In both cases, the screen is cleared as a side effect of the command.

The *display* parameter selects the active video adapter and mode or reconfigures the current adapter. If a display adapter that is not available is specified, MODE displays an error message.

The *shift* parameter is simply the single character R or L preceded by a comma. Each shift command causes the screen image to be shifted by two characters if the display adapter is in 80-column mode or by one character if it is in 40-column mode. When the T option is

also included in the command line, the screen image is shifted, a test pattern is displayed, and the user is prompted to indicate whether the screen should be shifted again. Note that use of *shift* causes part of the MODE program to become permanently resident in memory.

## Examples

In a system with both a color/graphics adapter and a monochrome display adapter, to select the monochrome display as the active display, type

```
C>MODE MONO  <Enter>
```

To select a color 80-column text mode on the color/graphics adapter, shift the screen image two characters to the left, and display a test pattern, type

```
C>MODE CO80,L,T  <Enter>
```

## Messages

**DOS 2.0 or later required**
MODE does not work with versions of MS-DOS earlier than 2.0.

**Do you see the leftmost 0? (Y/N)**

or

**Do you see the rightmost 9? (Y/N)**
When the *shift* and T options are used together, this message allows the user to shift the test-pattern display successive positions until it is properly centered.

**Incorrect DOS version**
The version of MODE is not compatible with the version of MS-DOS that is running.

**INTERNAL ERROR in MODE application**
An internal error occurred in the MODE utility and the requested reconfiguration was not carried out.

**Invalid parameter**
The specified display adapter or mode is not available.

**Requested Screen Shift out of range**
The display cannot be shifted any further.

**Unable to shift Screen left**
The screen has already been shifted as far left as possible or the active display adapter cannot be shifted (monochrome or enhanced graphics adapter).

**Unable to shift Screen right**
The screen has already been shifted as far right as possible or the active display adapter cannot be shifted (monochrome or enhanced graphics adapter).

# MODE

3.2

Configure Serial Port

External

## Purpose

Controls the configuration of the serial communications adapter. This form of the MODE command is included with PC-DOS beginning with version 1.1.

## Syntax

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

where:

| | |
|---|---|
| COM*n* | is the serial port (*n* = 1 or 2). |
| *baud* | is the baud rate (110, 150, 300, 1200, 2400, 4800, or 9600). |
| *parity* | is the type of parity checking (N = none, O = odd, E = even, default = E). |
| *databits* | is the number of bits per character (7 or 8, default = 7). |
| *stopbits* | is the number of stop bits (1 or 2, default = 1, except with 110 baud where default = 2). |
| P | causes continuous retries when the output device is not ready. |

## Description

This form of the MODE command configures the specified serial port for communication with an external device such as a printer, a terminal, or a modem.

The first parameter, COM*n*, designates the serial port to be configured (COM1 or COM2). Except for the port number and the baud rate, which are required, a parameter can be left unchanged by entering a comma without a value in its position in the command line. (If *all* optional parameters are to be left unchanged and P is not used in the command line, no commas are required.)

The baud rate must be one of the values 110, 150, 300, 600, 1200, 2400, 4800, or 9600. The first two digits can be used as an abbreviation for the full value.

The *parity* parameter specifies the type of parity checking to be done on each character and must be one of the characters N, O, or E (for none, odd, or even, respectively); the default is even parity. The *databits* parameter specifies the length of a character and must be either 7 or 8; the default is 7. The *stopbits* parameter is either 1 or 2. If *baud* is set for 110, the default number of *stopbits* is 2; otherwise, the default is 1.

The last parameter in the command line, P, configures the system to retry output continuously (or until Ctrl-Break is pressed) if the device interfaced to the serial port is not ready or not on line, rather than display an error message. Use of the P option causes part of the MODE program to become permanently resident in memory.

Consult the user's manual for the specific printer, modem, terminal, or other device to determine the proper settings for the MODE parameters.

If a serial printer is to be used instead of LPT1 as the system's default printer, the Redirect Printing MODE command must be specified *after* the Configure Serial Port MODE command.

## Example

To configure the first serial port for 9600 baud, no parity, 8 databits, and 1 stop bit, type

```
C>MODE COM1:9600,N,8,1  <Enter>
```

## Messages

**COM*n: baud, parity, databits, stopbits, timeout***
After the serial port is configured successfully, MODE displays an advisory message confirming the settings. If the P option was not used in the command line, a hyphen character (-) is displayed for *timeout*, to indicate no continuous retries if the printer is not ready or is not on line.

**COM port does not exist**
The serial port specified in the command line does not exist in the system.

**DOS 2.0 or later required**
MODE does not work with versions of MS-DOS earlier than 2.0.

**Incorrect DOS version**
The version of MODE is not compatible with the version of MS-DOS that is running.

**INTERNAL ERROR in MODE application**
An internal error occurred in the MODE utility and the requested reconfiguration was not carried out.

**Invalid baud rate specified**
The baud rate included in the command line was not one of the allowed values or was abbreviated incorrectly.

**Invalid parameters**
The command line specified a COM port that does not exist in the system or one of the configuration parameters for the COM port was not valid.

**No COM: ports**
The computer does not have any serial ports installed.

**Resident portion of MODE loaded**
The P option was specified in the command line and part of the MODE command has become permanently resident in memory, decreasing slightly the amount of memory available to other programs.

# MODE

3.2

Redirect Printing

External

## Purpose

Redirects output from a parallel port to a serial communications port. This form of the MODE command is included with PC-DOS beginning with version 1.1.

## Syntax

MODE LPT*n*[:][=COM*n*[:]]

where:

| | |
|---|---|
| LPT*n* | is the parallel port to be redirected (*n* = 1, 2, or 3). |
| COM*n* | is the serial port (*n* = 1 or 2) to be used for output instead of LPT*n*. |

## Description

This form of the MODE command redirects any output for the specified parallel port, sending it to the specified serial communications port instead. The parallel port can be LPT1, LPT2, or LPT3; the serial port can be either COM1 or COM2. A Configure Serial Port MODE command is required *before* the Redirect Printing MODE command, to configure the serial port for the proper baud rate, parity, word length, and stop bits.

Redirection can be canceled by entering *MODE LPT*n alone.

Use of MODE to redirect printer output causes part of the MODE program to become permanently resident in memory. Canceling the redirection will not remove this resident portion from memory.

## Example

To cause all output to the first parallel port (LPT1) to be redirected to the first serial port (COM1), type

```
C>MODE LPT1:=COM1:  <Enter>
```

## Messages

**DOS 2.0 or later required**
MODE does not work with versions of MS-DOS earlier than 2.0.

**Illegal device name**
Either the parallel port or the serial port specified in the command line does not exist in the system.

**Incorrect DOS version**
The version of MODE is not compatible with the version of MS-DOS that is running.

**INTERNAL ERROR in MODE application**
An internal error occurred in the MODE utility and the requested reconfiguration was not carried out.

**LPT*n*: not redirected**
No serial port was specified and any previous redirection from the specified parallel port was canceled.

**LPT*n*: redirected to COM*n*:**
The MODE command has successfully redirected the output for the specified parallel port to the specified serial port.

**Resident portion of MODE loaded**
Part of the MODE command has become permanently resident in memory, decreasing slightly the amount of memory available to other programs.

# MORE

2.0 and later

Display by Screenful

External

## Purpose

Displays output one screenful at a time on standard output.

## Syntax

MORE

## Description

The MORE filter reads lines of text from standard input and sends them to standard output one screenful (23 lines) at a time. At the end of each screenful, MORE displays the message -- *More* -- and then waits for any key to be pressed before it continues. (Pressing Crtl-C or Ctrl-Break terminates the MORE filter.)

The default input device is the keyboard; the default output device is the video display. Because standard input can be redirected, the MORE filter can also accept input from another character device or a file or from the piped output of another program or filter. Similarly, the output of MORE can be redirected to any character device or file or can be piped to another program (however, the message -- *More* -- will be included with the redirected or piped output).

## Examples

To display the file SHELL.C one screenful at a time, type

```
C>MORE < SHELL.C  <Enter>
```

To display the directory of \MASM\SOURCE in the current drive one screenful at a time, pipe the output of the DIR command to the MORE filter by typing

```
C>DIR \MASM\SOURCE | MORE  <Enter>
```

## Messages

**-- More --**
This informational message is displayed at the end of each screenful of text. Press any key to resume output.

**MORE: Incorrect DOS version**
The version of MORE is not compatible with the version of MS-DOS that is running.

# PATH

Define Command Search Path

2.0 and later

Internal

## Purpose

Specifies one or more additional drives and/or directories to be searched for a program or batch file if the file cannot be found in the current or specified drive and directory.

## Syntax

PATH [*drive*:][*path*][;[*drive*:][*path*]...]

or

PATH ;

where:

| | |
|---|---|
| *drive* | is the drive containing the disk to be searched for the executable file. |
| *path* | is the name of the directory to be searched for the executable file. |

## Description

When a command line is entered at the MS-DOS system prompt, the command processor first checks to see if the specified command is one of its internal commands. If it is not, the command processor searches the current directory of the current drive for a file with the same name and the extension .COM, .EXE, or .BAT, in that order. If found, the file is loaded into memory and executed (if the extension is .COM or .EXE) or interpreted by the resident batch-file processor (if the extension is .BAT); otherwise, MS-DOS displays the message *Bad command or file name*, followed by the system prompt. In versions 3.0 and later, a path can precede the command name, causing MS-DOS to make the initial search for a program or batch file under the specified path.

The PATH command designates one or more disk drives and/or directory paths to be searched sequentially for a program or batch file if the file cannot be found in the current or specified drive and directory. The drives and/or directory paths are searched in the order they appear in the PATH command. Multiple *drive*:*path* pairs can be specified, separated by semicolons. A copy of the PATH string is passed to each executing process as a part of the process's environment.

If the *drive* parameter is specified without an associated path, MS-DOS assumes the root directory of *drive*. If the PATH command is followed only by a semicolon, MS-DOS deletes the existing path. If the PATH command is entered with no parameters, MS-DOS displays the existing path.

Invalid or nonexistent drives and/or paths in the PATH command do not result in an error message but are ignored when the PATH string is inspected later during a search for a program or batch file.

The PATH command is generally placed in the AUTOEXEC.BAT file on the system disk so that the search order will be defined each time the system is turned on or restarted.

## Examples

To define the directory \BIN on the disk in drive A as the directory to be searched for a program or batch file if the file is not found in the current or specified directory, type

```
C>PATH A:\BIN  <Enter>
```

Subsequent entry of the command

```
C>PATH  <Enter>
```

results in the display

```
PATH=A:\BIN
```

To define the root, \BIN, \DOS, and \DATA directories on drive C and the \UTIL directory on the disk in drive B as the locations to be searched for a program or batch file if the file is not found in the current or specified directory, type

```
C>PATH C:\;C:\BIN;C:\DOS;C:\DATA;B:\UTIL  <Enter>
```

To delete the current search path, type

```
C>PATH ;  <Enter>
```

## Message

**No Path**
The PATH command was entered without parameters and no search path is currently in effect.

# PRINT

Print Spooler

2.0 and later

External

## Purpose

Loads and configures the background print spooler or adds or deletes files from the print spooler's queue.

## Syntax

PRINT [/D:*device*] [/B:*n*] [/M:*n*] [/Q:*n*] [/S:*n*] [/U:*n*] [[*drive*:][*path*]*filename*] [/C][/P] [[[*drive*:][*path*]*filename*] [/C][/P]...]

or

PRINT /T

where:

| | |
|---|---|
| *filename* | is the name of the file to be added to or deleted from the print queue, optionally preceded by a drive (and a path with versions 3.0 and later); wildcard characters are permitted. |
| /B:*n* | sets the print-buffer size in bytes (1–32767, default = 512) (versions 3.0 and later). |
| /C | deletes the immediately preceding file and all subsequent files from the print queue (until a /P switch is encountered). |
| /D:*device* | is the character device to be used for printing (default = PRN); must be the first switch, if used (versions 3.0 and later). |
| /M:*n* | is the length of time in timer ticks that PRINT keeps control during each of its time slices (1–255, default = 2) (versions 3.0 and later). |
| /P | adds the immediately preceding file and all subsequent files to the print queue (until a /C switch is encountered). |
| /Q:*n* | is the maximum number of files allowed in the print queue (1–32, default = 10) (versions 3.0 and later). |
| /S:*n* | is the number of time slices per second that PRINT gives control to the foreground process (1–255, default = 8) (versions 3.0 and later). |
| /T | terminates printing and empties the print queue. |
| /U:*n* | is the number of timer ticks that PRINT waits for a busy or unavailable printer or for a disk access or MS-DOS function call to terminate before giving up the time slice (1–255, default = 1) (versions 3.0 and later). |

## Description

The PRINT utility is a terminate-and-stay-resident (TSR) program that can print files from disk while other programs are running. PRINT maintains a first-in, first-out (FIFO) queue that can hold the names of as many as 32 files. PRINT does not attempt to interpret the contents of a file, except to expand tab characters (ASCII code 09H) with spaces to the next eight-column boundary and to interpret 1AH characters as end-of-file marks. (A program such as PRINT that can transfer files to a printer without any special knowledge of their contents or origin is called a print spooler.)

***Note:*** The PRINT utility continues printing a file until it encounters an end-of-file character (1AH). Therefore, if PRINT is used with nontext files, it may encounter a 1AH character before reaching the end of the file and terminate printing before the entire file has been processed. In such cases, files should be printed using the COPY command, with PRN as the destination.

The PRINT program employs a technique called time-slicing, which is based on its use of the timer-tick interrupt and its detailed knowledge of MS-DOS. PRINT uses this interrupt, which occurs 18.2 times per second on IBM PC-compatible machines, to divide the processor's time between an application or utility program (such as a word processor or a spreadsheet) and the print spooler. Because the application program typically controls the display screen and the keyboard and receives most of the CPU time, it is called the foreground program. The print spooler, which receives a lesser part of the CPU time and usually operates without indicating its status or progress to the operator, is called the background program.

The /B:*n*, /D:*device*, /Q:*n*, /M:*n*, /S:*n*, and /U:*n* switches configure the PRINT utility. These switches are used only the first time the PRINT command is entered after the system has been turned on or restarted.

The /D:*device* switch, which must be the first switch in the command line if used, specifies the peripheral device the print spooler is to use for output. This can be any legal character-output device that is present in the system. If /D:*device* is not included in the first PRINT command, PRINT prompts the user to select an output device (default = PRN). Once an output device has been assigned, a new device cannot be selected without restarting the system.

The /B:*n* switch sets the size of PRINT's file buffer, which controls the amount of data that is read from a file at one time for printing. The value of *n* must be between 1 and 32767 bytes (default value = 512). Large file buffers reduce the amount of extra disk activity caused by the print spooler, but they also reduce the amount of memory available for use by other programs. The /Q:*n* switch controls the size of PRINT's queue—that is, the number of files that can be held in the buffer pending printing. The queue can be configured to hold 1 to 32 files (default = 10).

The /S:*n*, /M:*n*, and /U:*n* switches, available only with versions 3.0 and later, control the time-slicing behavior of PRINT. The /S:*n* switch sets the number of time slices per second—that is, how many times per second—PRINT will be given control; *n* is in the

range 1 through 255 (default = 8). The /M:*n* switch sets the length of time (in timer ticks) that PRINT will keep control during each of its time slices; *n* is in the range 1 through 255 (default = 2). The /U:*n* switch specifies how long (in timer ticks) PRINT should wait for a busy or unavailable printer or for a disk access or MS-DOS function call to terminate before giving up its time slice; again, *n* is in the range 1 through 255 (default = 1). Unless there are special circumstances, the default values for these switches will give acceptable performance.

Files are added to the print queue by entering PRINT followed by one or more pathnames. Files are printed in the order they are placed in the queue. At the end of each file, the print spooler advances the paper to the top of the next page. If a filename containing wildcards is used, all matching files are added to the queue in the order in which they appear in the directory. After a file is queued for printing, it should not be renamed or erased, nor should the disk containing the file be removed, until the printing is complete.

***Note:*** Each print queue entry can be a maximum of 63 characters, including the drive and path.

The /P and /C switches allow files to be added to and deleted from the print queue in the same command line. The /P switch (the default) adds to the print queue the immediately preceding file in the command line and all subsequent files until a /C switch is encountered. Conversely, the /C switch cancels printing for the immediately preceding file in the command line and for all subsequent files until a /P switch is encountered. If a canceled file is currently being printed, PRINT prints the message *File* filename *canceled by operator* on the listing, sounds the printer's alarm (if it has one), and advances the paper to the top of the next page.

The /T switch terminates printing by deleting all files from the print queue. If a file is currently being printed, PRINT prints the message *All files canceled by operator* on the listing, sounds the printer's alarm (if it has one), and advances the paper to the top of the next page.

If PRINT encounters a disk error while attempting to print a particular file, it cancels that file, prints an error message on the printer, sounds the printer's alarm (if it has one), advances the paper to the top of the next page, and goes to the next file in the print queue.

If the PRINT command is entered with no parameters, the contents of the print queue are displayed.

Because PRINT is a TSR utility, it reduces the amount of memory available for use by other programs. The only way to recover the memory occupied by PRINT, even after printing is complete, is to restart the system.

## Examples

To install and configure the PRINT program and specify the auxiliary device (AUX) as the printing device, with a print queue that can hold as many as 32 filenames and with a buffer size of 2048 bytes, type

```
C>PRINT /D:AUX /Q:32 /B:2048  <Enter>
```

To add the file DOC.TXT in the current directory of the current drive to the print spooler's queue, type

```
C>PRINT DOC.TXT  <Enter>
```

To delete the file READY.TXT from the print queue and simultaneously add the files FINAL.TXT and REPORT.TXT to the queue, type

```
C>PRINT READY.TXT /C FINAL.TXT /P REPORT.TXT  <Enter>
```

To cancel the file being printed and remove all pending files from the print queue, type

```
C>PRINT /T  <Enter>
```

## Messages

***filename* File not found**
A disk was changed or the file was renamed or erased after the PRINT command was entered but before the file was actually printed.

***filename* File not in print queue**
A command line with a /C switch specified a file that is not in the print queue.

***filename* is currently being printed**
This informational message shows which file PRINT is currently printing.

***filename* is in queue**
This informational message shows which file is in the queue waiting to be printed.

***filename* Pathname too long**
The pathname of a file to be printed exceeded 63 characters.

**Access denied**
An attempt was made to print a locked file.

**All files canceled by operator**
The /T switch was included in the command line. PRINT terminates printing of the current file, empties the print queue, sounds the printer alarm (if it has one), and advances the paper to the top of the next page.

**Cannot use PRINT - Use NET PRINT**
If network support has been installed, the NET PRINT command must be used to print files.

**Errors on list device indicate that it may be off-line. Please check it.**
The printer has been turned off or placed off line while files are still in the print queue.

**File *filename* canceled by operator**
A PRINT command was entered with the /C switch to cancel a specific file. If the specified file is currently being printed, PRINT terminates printing of the file, sounds the printer alarm (if it has one), advances the paper to the top of the next page, and resumes printing with the next file in the queue.

**Incorrect DOS version**
The version of PRINT is not compatible with the version of MS-DOS that is running.

**Invalid drive specification**
A drive letter specified in the command line is invalid or does not exist in the system.

**Invalid parameter**
The command line included an invalid switch or configuration switches were used after the first time the PRINT command was used.

**List output is not assigned to a device**
An invalid destination device was previously entered. Restart the system and specify a valid device in the PRINT command.

**Name of list device [PRN]:**
This message is displayed in response to the first PRINT command line if the /D:*device* switch was not included. Specify any valid character-output device (default = PRN).

**No paper error writing device *device***
An out-of-paper device error was detected while printing on the specified device.

**PRINT queue is empty**
No files are waiting to be printed.

**PRINT queue is full**
No additional files can be added to the print queue until the current file is printed. To increase the size of the print queue, restart the system and use the /Q:*n* switch in the PRINT command.

**Resident part of PRINT installed**
This informational message is displayed on the first entry of a PRINT command to indicate that the PRINT utility is now resident in memory. The amount of memory available to application programs is reduced accordingly.

# PROMPT

2.0 and later

Define System Prompt

Internal

## Purpose

Defines the form of the command processor's prompt. This command is included in PC-DOS beginning with version 2.1.

## Syntax

PROMPT [*string*]

where:

*string* is a combination of ordinary printable characters and the following special display codes:

| Code | Meaning |
|---|---|
| $b | ¦ character |
| $d | Current date (in the form *Day mm-dd-yyyy*) |
| $e | Escape character (1BH) |
| $g | > character |
| $h | Backspace character (erases the previous character) |
| $l | < character |
| $n | Current drive |
| $p | Current drive and path |
| $q | = character |
| $t | Current time (in the form *hh:mm:ss.hh*) |
| $v | MS-DOS version number |
| $_ | Carriage return/linefeed pair (starts a new line) |
| $$ | $ character |

## Description

The system's default command processor, COMMAND.COM, displays a prompt on the screen whenever it is ready to accept a command from the user. The command processor determines the format of the prompt from the PROMPT environment variable, if it exists. Otherwise, it uses the default format, which in most OEM implementations of MS-DOS is the letter of the current drive followed by a greater-than sign (for example, C>).

The PROMPT command allows the user to customize the system prompt. This command is usually included in the AUTOEXEC.BAT file so that MS-DOS displays the custom prompt when the system is turned on or restarted.

The *string* parameter can be any combination of printable characters and the special $ control codes listed in the preceding table. The special $ codes allow certain variable information, such as the date and time, to be obtained from the operating system and displayed as part of the prompt. Such system information can be edited in the prompt with the backspace function, which is invoked with the code $h.

***Note:*** When the time is displayed as part of a prompt, it is updated only when the command processor redisplays the prompt.

The escape character, invoked with the code $e, can be used to include standard ANSI escape sequences in *string* to control the appearance of text or its position on the screen. *See* USER COMMANDS: ANSI.SYS for further information on the ANSI escape sequences and the ANSI device driver.

If PROMPT is entered with no parameters, the system prompt is reset to the default format.

The PROMPT command works by modifying the PROMPT environment variable. The same result can be obtained using the SET command with *PROMPT*=string as its argument. *See* USER COMMANDS: SET for further discussion of the environment block and environment variables.

## Examples

To define the system prompt as the word *Command* followed by a colon, type

```
C>PROMPT Command:  <Enter>
```

On fixed-disk-based systems it is desirable to display the current drive and path as part of the prompt. To define such a prompt followed by a > character, type

```
C>PROMPT $p$g  <Enter>
```

To define the system prompt to display the time, date, and current drive and path followed by a > character, each on a separate line, type

```
C>PROMPT $t$_$d$_$p$g  <Enter>
```

The system will respond with a display in the following form:

```
16:07:31.56
Thu  6-18-1987
C:\BIN\DOS>
```

To create a prompt that displays the time without the seconds and hundredths of a second, followed by a space and the date without the year, followed by a space and the current drive and a > character, type

```
C>PROMPT $t$h$h$h$h$h $d$h$h$h$h$h $n$g  <Enter>
```

The system will respond with

```
16:07 Thu  6-18 C>
```

To define a prompt that always displays the current time and date in the upper right corner of the screen before displaying the current drive and the > character on the current line, type

```
C>PROMPT $e[s$e[0;60H$t$h$h$h$h$h$h $d$e[u$n$g  <Enter>
```

The escape sequence *$e[s* saves the current cursor position; the sequence *$e[0;60H* positions the cursor at row 0, column 60; the next several codes format the date and time; the sequence *$e[u* restores the original cursor position. (This example requires that the ANSI driver be loaded to interpret the escape sequences.)

# RAMDRIVE.SYS

3.2

Virtual Disk

External

## Purpose

Creates a virtual disk in memory.

## Syntax

DEVICE=[*drive*:][*path*]RAMDRIVE.SYS [*size*] [*sector*] [*directory*] [/A ¦ /E]

where:

| | |
|---|---|
| *size* | is the size of the virtual disk in kilobytes (minimum = 16, default = 64). |
| *sector* | is the sector size in bytes (128, 256, 512, or 1024; default = 128). |
| *directory* | is the maximum number of entries in the virtual disk's root directory (3–1024, default = 64). |
| /A | causes RAMDRIVE to use Lotus/Intel/Microsoft Expanded Memory for storage (cannot be used with /E). |
| /E | causes RAMDRIVE to use extended memory for storage (cannot be used with /A). |

***Note:*** Unless a /A or /E switch is used, the virtual disk is created in conventional memory.

## Description

The RAMDRIVE.SYS installable device driver allows the configuration of one or more virtual disks (sometimes referred to as electronic disks or RAMdisks). A virtual disk is implemented by mapping a disk's structure — directory, file allocation table, and files area — onto an area of random-access memory, rather than onto actual sectors located on a magnetic recording medium. Access to files stored on a virtual disk is very fast, because no moving parts are involved and the "disk" operates at the speed of the system's memory.

***Warning:*** Because a RAMdisk resides entirely in RAM and is therefore volatile, any information stored there is irretrievably lost when the computer loses power or is restarted.

RAMDRIVE.SYS can create a virtual disk in conventional memory, extended memory, or Lotus/Intel/Microsoft Expanded Memory. Conventional memory is the term for the up-to-640 KB of RAM that contain MS-DOS and any application programs. Extended memory is the term for the memory at addresses above 1 MB (100000H) that is available on 80286-based personal computers such as the IBM PC/AT. Expanded memory is the term for a subsystem of bank-switched memory boards (and a driver to manage them) that is compatible with the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS).

A virtual disk can be installed in conventional memory by simply inserting the line *DEVICE=RAMDRIVE.SYS* into the system's CONFIG.SYS file and restarting the system. A

new "drive" then becomes available in the system, with a default size of 64 KB, 128-byte sectors, and 64 available directory entries (assuming memory is sufficient). The virtual disk is assigned the next available drive letter (which is displayed in RAMDRIVE's sign-on message). The drive letter assigned depends on the number of other physical and virtual disks in the system and also on the position of the *DEVICE=RAMDRIVE.SYS* line in the CONFIG.SYS file relative to other installed block devices. Available memory permitting, multiple virtual disks can be created by using multiple *DEVICE=RAMDRIVE.SYS* lines. Several optional parameters allow the user to customize the size and configuration of the virtual disk and to use extended memory or expanded memory if it is available.

The *size* parameter specifies the amount of RAM, in kilobytes, to be allocated to the virtual disk. The default is 64 KB, but any size from 16 KB to the total amount of available memory can be specified.

The *sector* parameter sets the virtual sector size used within the virtual disk. The *sector* value can be 128, 256, 512, or 1024 bytes (default = 128 bytes). Selection of the smallest sector size results in a minimum of wasted virtual disk space per file but also results in a somewhat slower transfer of data. Physical disk devices on IBM PC-compatible systems always use 512-byte sectors.

***Warning:*** The 1024-byte sector size is not supported in most implementations of MS-DOS and will terminate the installation of RAMDRIVE.SYS if it is used. Check the documentation included with the computer to see if this value is supported.

The *directory* parameter sets the number of available entries in the virtual disk's root directory. The allowed range is 3 to 1024 (default = 64). Each directory entry requires 32 bytes. RAMDRIVE rounds the number of available directory entries up, if necessary, so that an integral number of sectors are assigned to the root directory.

The /A switch causes Lotus/Intel/Microsoft Expanded Memory to be used for the virtual disk, rather than conventional memory; the /E switch causes extended memory to be used. Either option allows very large virtual disks to be configured while still leaving the maximum amount of conventional memory available for use by application programs. The /A and /E switches cannot be used together.

***Note:*** If RAMDRIVE uses conventional memory for virtual disk storage, the memory cannot be reclaimed except by modifying the CONFIG.SYS file and restarting the system.

## Examples

To create a virtual disk drive with the default values of 64 KB disk size, 128-byte sectors, and 64 available directory entries, include the following command

```
DEVICE=RAMDRIVE.SYS
```

in the CONFIG.SYS file and restart the system.

To create a 4 MB virtual disk drive in Lotus/Intel/Microsoft Expanded Memory, with 512-byte sectors and 224 available directory entries, when RAMDRIVE.SYS is located in a directory named \DRIVERS on drive C, include the command

```
DEVICE=C:\DRIVERS\RAMDRIVE.SYS 4096 512 224 /A
```

in the CONFIG.SYS file and restart the system.

## Messages

**Microsoft RAMDrive version *n.nn* virtual disk *X*:**
**Disk size: *nn*k**
**Sector size: *nnn* bytes**
**Allocation unit: *n* sectors**
**Directory entries: *nnn***

RAMDRIVE.SYS was successfully installed and this message informs the user of the version of RAMDRIVE.SYS that created the virtual disk, the drive letter assigned to the disk, and the characteristics of the disk.

**RAMDrive: Above Board Memory Manager not present**
The /A switch was used in the command line and the Lotus/Intel/Microsoft Expanded Memory Manager is not present in the system. Place the DEVICE command that loads the memory manager *before* the *DEVICE=RAMDRIVE.SYS* command in the CONFIG.SYS file.

**RAMDrive: Above Board Memory Status shows errors**
The Above Board device driver is bad or damaged or the board itself is defective. Consult the Above Board manual or the manufacturer.

**RAMDrive: Computer must be PC-AT, or PC-AT compatible.**
The /E switch was used in the command line and the computer is not an 80286-based IBM PC/AT or compatible.

**RAMDrive: Incorrect DOS version**
The version of RAMDRIVE.SYS is not compatible with the version of MS-DOS that is running.

**RAMDrive: Insufficient memory**
Available memory is insufficient for RAMDRIVE.SYS to create a virtual drive.

**RAMDrive: Invalid parameter**
One of the parameters supplied in the command line is incorrect or is not supported by the computer.

**RAMDrive: I/O error accessing drive memory**
The Expanded Memory Manager device driver is bad or damaged or the board itself is defective. Consult the board's manual or contact the manufacturer.

**RAMDrive: No extended memory available**
The /E switch was specified but the system does not contain extended memory.

# RECOVER

2.0 and later

Recover Files

External No Net

## Purpose

Reconstructs files from a disk that has developed unreadable sectors or has a damaged directory.

## Syntax

RECOVER *drive*:

or

RECOVER [*drive*:][*path*]*filename*

where:

| | |
|---|---|
| *drive* | is the letter of the drive holding the disk with a damaged directory. |
| *filename* | is the name of the file that will be reconstructed, optionally preceded by a drive and/or path; wildcard characters are not permitted. |

## Description

The RECOVER command partially rescues a file on a disk that has developed bad sectors by deleting the bad sectors from the file. RECOVER can also reconstruct files (including files stored in subdirectories) from a disk that has a damaged directory.

When RECOVER is used with a filename, the file is read allocation unit by allocation unit; unreadable allocation units are marked as bad and are no longer allocated to the file. The resulting file is usable, although the data contained in the bad allocation units is lost. (The recovered file may or may not be reusable by the specific application that created it.) The directory entry for *filename* is also adjusted to reflect the sectors that were lost and the bad sectors are marked in the disk's file allocation table so that they are not reused for another file.

If a disk's directory is damaged, it still may be possible to recover all the files on the disk and build a new directory by using RECOVER with *drive* as the only command-line parameter. RECOVER completely erases the previous contents of the damaged directory and constructs new directory entries for each of the original files by inspecting the disk's file allocation table. The recovered files receive names of the form FILE*nnnn*.REC, starting with FILE0001.REC. Each recovered file's size is always a multiple of the disk cluster size, so recovered files may require editing to eliminate spurious data at the ends of the files.

RECOVER restores each subdirectory as an individual file that contains the names of the files originally stored in it. The actual files contained within those subdirectories are also reconstructed, although they are no longer associated with the subdirectory in which they

originally resided. Restored files and subdirectories, regardless of their location on the damaged disk, are placed in the new root directory. If there are more files on the damaged disk than can be contained in the new root directory (for example, more than 112 for a 5.25-inch, 360 KB floppy disk), the user must repeat the RECOVER command after copying the already-recovered files to another disk and deleting them from the damaged disk.

## Examples

To recover the file MENUMGR.C in the current directory of the current drive, type

```
C>RECOVER MENUMGR.C  <Enter>
```

To recover all files on the disk in drive B, which has a damaged directory, type

```
C>RECOVER B:  <Enter>
```

## Messages

***n* file(s) recovered**
When RECOVER is used on a disk with a damaged directory, this informational message is displayed at the conclusion of processing to indicate how many files of the form FILE*nnnn*.REC were constructed.

***n* of *n* bytes recovered**
When RECOVER is used on a damaged file, this informational message is displayed at the conclusion of processing to advise how many bytes of the file were recovered.

**Cannot RECOVER a Network drive**
Files on a drive assigned to a network cannot be recovered.

**File not found**
The file specified in the command line cannot be found or does not exist.

**Incorrect DOS version**
The version of RECOVER is not compatible with the version of MS-DOS that is running.

**Invalid drive or file name**
An invalid drive letter was specified or the filename contains a wildcard.

**Invalid number of parameters**
More than one drive letter or filename was specified in the command line.

**Press any key to begin recovery of the file(s) on drive *X***
This prompt message gives the user the opportunity to change disks after the RECOVER program is loaded but before processing begins.

**Warning - directory full**
New directory entries for the reconstructed files cannot be created because the root directory is full. Copy the recovered files to another disk, delete them from the damaged disk, and then repeat the RECOVER command on the damaged disk.

# RENAME or REN

1.0 and later

Change Filename

Internal

## Purpose

Changes the name of a file or set of files.

## Syntax

RENAME [*drive*:][*path*]*oldname newname*

or

REN [*drive*:][*path*]*oldname newname*

where:

| | |
|---|---|
| *oldname* | is the name of an existing file or set of files, optionally preceded by a drive and/or path; wildcard characters are permitted. |
| *newname* | is the new name to be assigned to *oldname*; wildcard characters are permitted, but a drive and/or path cannot be specified. |

## Description

The RENAME command changes the name of an existing file or set of files. It does not make copies of files or move files from one location in the disk's directory structure to another or from one drive to another.

The *oldname* parameter can refer to a single file or can include wildcards to specify a set of files; a drive and path can be included as part of *oldname*.

The *newname* parameter specifies the new name to be given to the file or files; it cannot include a drive or path. A wildcard in *newname* causes that portion of the original filename to be left unchanged. If the new name for a file is the same as the name of an existing file, RENAME terminates with an error message.

## Examples

To rename the file REVS.DOC, located in the current directory of the current drive, to CHANGES.TXT, type

```
C>RENAME REVS.DOC CHANGES.TXT  <Enter>
```

or

```
C>REN REVS.DOC CHANGES.TXT  <Enter>
```

To rename all files with a .DOC extension in the \SOURCE directory on the disk in drive D to have a .TXT extension, type

```
C>REN D:\SOURCE\*.DOC *.TXT  <Enter>
```

## Messages

**Duplicate file name or File not found**
The new name specified for a file already exists or a file with the old name cannot be found or does not exist.

**Invalid directory**
The command line included a reference to a directory that is invalid or does not exist.

**Invalid drive specification**
The command line included a reference to a disk drive that is invalid or does not exist in the system.

**Invalid number of parameters**
The command line included too few or too many filenames.

**Invalid parameter**
The *newname* parameter in the command line included a drive and/or path.

# REPLACE

3.2

Update Files

External

## Purpose

Selectively adds or replaces files on a disk.

## Syntax

REPLACE [*drive*:]*pathname* [*drive*:][*path*] [/A][/D][/P][/R][/S][/W]

where:

| | |
|---|---|
| *pathname* | is the name and location of the source files to be transferred, optionally preceded by a drive; wildcard characters are permitted in the filename. |
| *drive:path* | is the destination for the file being transferred; filenames are not permitted in the destination parameter. |
| /A | transfers only those source files that do not exist at the destination (cannot be used with /S or /D). |
| /D | transfers only those source files with a more recent date than their destination counterparts (cannot be used with /A). |
| /P | prompts the user for confirmation before each file is transferred. |
| /R | allows REPLACE to overwrite destination read-only files. |
| /S | searches all subdirectories of the destination directory for a match with the source files (cannot be used with /A). |
| /W | causes REPLACE to wait for the disk to be changed before transferring files. |

## Description

The REPLACE utility allows files to be updated easily to more recent versions. REPLACE examines the source and destination directories and, depending on the switches used in the command line, selectively updates matching files or copies only those files that exist on the source disk but not the destination disk.

The *pathname* parameter (the source) specifies the name and location of the files to be transferred (optionally preceded by a drive); wildcards are permitted in the filename. The *drive:path* parameter (the destination) specifies the location of the files to be replaced and can consist of a drive, a path, or both. If only a drive is specified as the destination, REPLACE assumes the current directory of the disk in that drive. If the destination is omitted completely, REPLACE assumes the current drive and directory. The /S switch causes REPLACE to also search all subdirectories of the destination directory for files to be replaced.

The /A, /D, and /P switches allow selective replacement of files on the destination disk. When the /A switch is used, REPLACE transfers only those files on the source disk that do not exist in the destination directory. When the /D switch is used, REPLACE transfers only

those source files that match the destination filenames but have a more recent date than their destination counterparts. (The /D switch is not available with the PC-DOS version of REPLACE.) The /P switch causes REPLACE to prompt the user for confirmation before each file is transferred.

The /R switch allows the replacement of read-only as well as normal files. If the /R switch is not used and one of the destination files that would otherwise be replaced is marked read-only, the REPLACE program terminates with an error message. (REPLACE cannot be used to update hidden or system files.)

The /W switch causes REPLACE to pause and wait for the user to press any key before beginning the transfer of files. This allows the user to change disks in floppy-disk systems with no fixed disk and in those cases where the REPLACE program itself is present on neither the source nor the destination disk.

## Return Codes

| | |
|---|---|
| 0 | The REPLACE operation was successful. |
| 1 | An error was found in the REPLACE command line. |
| 2 | No matching files were found to replace. |
| 3 | The source or destination path was invalid or does not exist. |
| 5 | One of the files to be replaced was marked read-only and the /R switch was not included in the command line. |
| 8 | Memory was insufficient to run the REPLACE command. |
| 15 | An invalid drive was specified in the command line. |
| *Other* | Standard MS-DOS error codes (returned on a failed Interrupt 21H file-function request). |

## Examples

To replace the files in the directory \SOURCE on the current drive with all matching files on the disk in drive A that have a more recent date, type

```
C>REPLACE A:*.* \SOURCE /D  <Enter>
```

To transfer from the disk in drive A only those files that are not already present in the current directory, type

```
C>REPLACE A:*.* /A  <Enter>
```

## Messages

### *n* File(s) added

After the replacement operation is completed, if the /A switch was used in the command line, REPLACE displays the total number of files added.

### *n* File(s) replaced

After the replacement operation is completed, REPLACE displays the total number of files processed.

**Access denied *'pathname'***
One of the files to be replaced on the destination disk is marked read-only and the /R switch was not included in the command line.

**Add *pathname*? (Y/N)**
The /A and /P switches were specified in the command line and REPLACE prompts the user for confirmation before adding each file.

**Adding *pathname***
The /A switch was specified in the command line and REPLACE displays the name of each file it adds.

**File cannot be copied onto itself *'pathname'***
The source and destination command-line parameters specified the same file in the same location.

**Incorrect DOS Version**
The version of REPLACE is not compatible with the version of MS-DOS that is running.

**Insufficient disk space**
The destination disk does not have enough available space to hold the files being added or replaced.

**Insufficient memory**
The system does not have enough RAM available to process the REPLACE command.

**Invalid drive specification *'X:'***
The command line specified a disk drive that is invalid or does not exist in the system.

**Invalid parameter *'switch'***
The command line included a switch that is not supported by the REPLACE command.

**No files added**
The /A switch was used and the specified file(s) already exist on the destination disk.

**No files found *'pathname'***
The files to be added or replaced on the destination disk were not found on the source disk.

**No files replaced**
The files at the destination are identical with the files on the source disk or do not meet the criteria specified by the switches.

**Parameters not compatible**
The command line included two or more switches that cannot be used together.

**Path not Found *'pathname'***
The source or destination parameter included a nonexistent path or directory.

**Path too long**
The source or destination parameter included a path element that is too large (probably because of a missing backslash character [\]).

**Press any key to begin adding file(s)**
The /W and /A switches were specified in the command line and REPLACE waits for the user to press a key before proceeding, allowing disks to be changed.

**Press any key to begin replacing file(s)**
The /W switch was specified in the command line and REPLACE waits for the user to press a key before proceeding, allowing disks to be changed.

**Replace *pathname*? (Y/N)**
The /P switch was specified in the command line and REPLACE prompts the user for confirmation before replacing the file.

**Replacing *pathname***
This informational message indicates the progress of the REPLACE command by displaying the name of each file as it is being replaced.

**Source path required**
Although the destination parameter can usually be omitted and defaults to the current drive and directory, the source location for the files to be replaced must always be specified.

**Unexpected DOS Error *n***
This message usually indicates a bad or damaged disk. Use the CHKDSK command to determine the problem.

# RESTORE

2.0 and later

Restore Backup Files

External

## Purpose

Restores files from a disk created with the BACKUP command.

## Syntax

RESTORE *drive1*: [*drive2*:][*pathname*] [/A:*date*] [/B:*date*] [/E:*time*] [/L:*time*][/M][/N] [/S][/P]

where:

| | |
|---|---|
| *drive1* | is the drive that contains the backup files created by the BACKUP command. |
| *drive2* | is the drive to which the backup files will be restored. |
| *pathname* | is the name of the file(s) to be restored from *drive1*; wildcard characters are permitted in the filename. If a path is used, a filename must be specified. |
| /A:*date* | restores files that were modified on or after *date*. |
| /B:*date* | restores files that were modified on or before *date*. |
| /E:*time* | restores files modified at or before *time*. |
| /L:*time* | restores files modified at or after *time*. |
| /M | restores only files modified since the last backup. |
| /N | restores only files that no longer exist on the destination disk. |
| /P | prompts the user for confirmation before restoring hidden or read-only files or before overwriting files that have changed since they were last backed up . |
| /S | restores all files in the subdirectories of the specified directory, in addition to the files in the specified directory. |

***Note:*** The PC-DOS version of RESTORE supports only the /P and /S switches.

## Description

The RESTORE command restores files from a backup disk or directory created with the BACKUP command to their original location in a directory structure. Before version 3.1, the RESTORE command could restore files only from one floppy disk to another or from a floppy disk to a fixed disk. With later versions, RESTORE can also restore files from one fixed disk to another or from a fixed disk to a floppy disk.

The *drive1* parameter specifies the source for the backed-up files. If the source disk is a fixed disk, the backup files are always obtained from the directory \BACKUP. If multiple floppy disks were used to hold the backed-up files, RESTORE prompts the user for each disk as it is required.

The destination can be any combination of a drive, a path, and a filename; the filename can include wildcards. If the destination drive is omitted, MS-DOS assumes the current drive. If a path is not specified, the files are restored to the current directory. (Note that files must be restored to the same directory they were backed up from.) If a path is specified, a filename must be specified as well. If neither a path nor a filename is included in the command line, all directories, subdirectories, and files on the backup disk(s) are restored to the destination disk. The /S switch can be used to force restoration of the files in all the subdirectories of a named directory.

Files are restored in the order they were backed up, regardless of their current order on the destination disk. If files with the same name and location already exist on the destination disk, they are replaced by the backup copies.

The RESTORE program supports a number of switches that allow selective restoration of files from the backup disk. The /A:*date*, /B:*date*, /E:*time*, and /L:*time* switches allow files to be restored based on the time and/or date they were backed up. The /M switch restores only those files that have been changed on the destination disk since the backup disk was created. The /P switch prompts the user before restoring a hidden or read-only file or a file that has been changed since it was last backed up.

The MS-DOS and PC-DOS RESTORE programs are compatible except when a /A:*date*, /B:*date*, /E:*time*, /L:*time*, /M, or /N switch is used. These switches are not supported in the PC-DOS version.

***Warning:*** The RESTORE command should not be used on a disk drive affected by an ASSIGN, SUBST, or JOIN command.

## Return Codes

0 The restore operation was successful.
1 No files were found to restore.
2 Some files were not restored because of a file-sharing conflict (versions 3.0 and later).
3 The restore operation was terminated by the user.
4 The program was terminated by an unrecoverable (critical) hardware error.

## Examples

To restore the file named MENUMGR.C from the backup disk in drive A to the directory named \SOURCE on the disk in drive B, type

```
C>RESTORE A: B:\SOURCE\MENUMGR.C  <Enter>
```

To restore all the files on the backup disk in drive A to their original locations in the directory structure of drive C, type

```
C>RESTORE A: C:\*.* /S  <Enter>
```

To restore all the files with the extension .C from the backup disk in drive A to the directory named \SOURCE on drive C, requesting confirmation for those files that are read-only or hidden, type

```
C>RESTORE A: C:\SOURCE\*.C /P  <Enter>
```

## Messages

***** Files were backed up at *time* on *date* *****
This informational message shows when the BACKUP command was used on the backed-up files.

***** Not able to restore file *****
The backup file or the destination disk contains an error. Use the CHKDSK command to determine the problem.

***** Restoring files from drive *X*: *****
**Diskette: *n***
This informational message indicates the progress of the RESTORE command.

**DOS 2.0 or later required**
RESTORE does not work with versions of MS-DOS earlier than 2.0.

**File creation error**
The destination directory is full. This usually occurs only if the destination is the root directory but can also happen if a file is being restored to a subdirectory and the disk itself is full.

**Incorrect DOS version**
The version of RESTORE is not compatible with the version of MS-DOS that is running.

**Insert backup diskette *n* in drive *X*:**
**Strike any key when ready**
This message prompts the user to insert the next backup disk in sequence. Disks used in multidisk backups should always be labeled and numbered during a BACKUP operation.

**Insert restore target diskette in drive *X*:**
**Strike any key when ready**
This prompt is displayed when files are being restored to a floppy disk.

**Insufficient memory**
Available memory is not sufficient for the RESTORE program to execute.

**Invalid drive specification**
The command line included a drive that is invalid or does not exist in the system.

**Invalid number of parameters**
The command line included too many or too few parameters.

**Invalid parameter**
The command line included an invalid switch or other parameter.

**Invalid path**
The destination parameter included a path that is invalid or does not exist.

**Restore file sequence error**
Files are being restored from a multidisk set of backup disks and a floppy disk was used out of order.

**Source and target drives are the same**
Files cannot be restored from a drive to the same drive.

**Source does not contain backup files**
The files on the backup disk are not in the special format used by the BACKUP and RESTORE programs.

**System files restored**
**Target disk may not be bootable**
The backup disk included copies of the hidden operating-system files MSDOS.SYS and IO.SYS (or IBMDOS.COM and IBMBIO.COM in PC-DOS) and these files were restored to the destination disk. The destination disk is bootable only if these two files are the first files on the disk and IO.SYS (or IBMBIO.COM) is written into contiguous clusters.

**Target is full**
The destination disk is full and no further files can be restored.

**Target is Non-Removable**
The disk to which files are being restored is not removable.

**The last file was not restored**
The destination disk is full or the last file on the backup disk was bad.

**Warning! Diskette is out of sequence**
**Replace diskette or continue if okay**
Files are being restored from a multidisk set of backup disks and a floppy disk was used out of order.

**Warning! File *filename***
**is a hidden file**
**Replace the file (Y/N)?**
The backed-up file has the same filename as a hidden file on the destination disk, which may be overwritten. (This message appears only if the /P switch was used.) Respond with *Y* to overwrite the file on the destination disk; respond with *N* to leave the destination file unchanged and continue the RESTORE operation.

**Warning! File *filename***
**is a read-only file**
**Replace the file (Y/N)?**
The backed-up file has the same name as a read-only file on the destination disk, which may be overwritten. (This message appears only if the /P switch was used.) Respond with

*Y* to overwrite the file on the destination disk; respond with *N* to leave the destination file unchanged and continue the RESTORE operation.

**Warning! File *filename***
**was changed after it was backed up**
**Replace the file (Y/N)?**
Data has been changed or added to the destination file since the backup disk was created and this data will be lost if the file is restored. (This message appears only if the /P switch was used.) Respond with *Y* to restore the backed-up file; respond with *N* to leave the destination file unchanged and continue the RESTORE operation.

**Warning! No files were found to restore**
No files were found on the backup disk that matched the destination file specification.

# RMDIR or RD

Remove Directory

2.0 and later

Internal

## Purpose

Removes an empty directory from the hierarchical file structure.

## Syntax

RMDIR [*drive*:][*path*]*directory_name*

or

RD [*drive*:][*path*]*directory_name*

where:

*directory_name* is the name of the directory to be removed, optionally preceded by a drive and/or path.

## Description

The RMDIR command removes an empty directory from a disk's hierarchical file structure. The directory being deleted cannot contain any files or subdirectories (except for the special . and .. entries). The root directory or current directory of a disk cannot be deleted.

If the *path* parameter is used, it must specify a valid existing path. If no path is specified and *directory_name* is not preceded by a backslash (\), MS-DOS assumes that the directory to be removed is a subdirectory of the current directory. If no path is specified and *directory_name* is preceded by a backslash, MS-DOS assumes that the directory is a subdirectory of the root directory. The length of the full path (including the drive designator and directory name) must not exceed 63 characters.

The RMDIR command should not be used to remove subdirectories from drives affected by an ASSIGN or JOIN command. A directory affected by the SUBST command cannot be removed.

***Note:*** If a directory contains files marked as hidden or system, that directory cannot be removed even though no files appear to exist when the directory contents are viewed using the DIR command.

## Example

To remove the empty directory \LIB, which is a subdirectory of the \MSC directory on the disk in drive A, type

```
C>RMDIR A:\MSC\LIB  <Enter>
```

or

```
C>RD A:\MSC\LIB  <Enter>
```

## Message

**Invalid path, not directory, or directory not empty**
The named directory cannot be deleted because it does not exist, some element of the path to the directory does not exist, or the directory contains files or subdirectories.

# SELECT

IBM

Configure System Disk for a Specific Country

External

## Purpose

Creates a system disk with time, date, and keyboard configured for a selected country. This command is available only with PC-DOS.

## Syntax

SELECT [[*drive1*:] *drive2*:[*path*]] *country keyboard*

where:

*drive1* is a floppy-disk drive (A or B) containing the distribution disk or, at a minimum, the PC-DOS system files, COMMAND.COM, and the FORMAT and XCOPY utilities (default = drive A) (version 3.2).

*drive2* is the drive containing the disk to receive the PC-DOS system files and country information and can include a path (default = drive B) (version 3.2).

*country* is a code from the table below that controls the time, date, and currency formats.

*keyboard* is a code from the table below that controls the keyboard configuration.

| Country | Country Code | Keyboard Code |
|---|---|---|
| Australia | 061 | * |
| Belgium | 032 | * |
| Canadian French | 002 | * |
| Denmark | 045 | * |
| Finland | 358 | * |
| France | 033 | FR |
| West Germany | 049 | GR |
| Israel | 972 | * |
| Italy | 039 | IT |
| Middle East | 785 | * |
| Netherlands | 031 | * |
| Norway | 047 | * |
| Portugal | 351 | * |
| Spain | 034 | SP |
| Sweden | 046 | * |
| Switzerland | 041 | * |

*(more)*

| Country | Country Code | Keyboard Code |
|---|---|---|
| United Kingdom | 044 | UK |
| United States | 001 | US |

*Available only in version 3.2 and may be supplied on a separate floppy disk.

## Description

The SELECT utility allows the user to create a bootable system disk configured for a particular country's keyboard layout and date, time, and currency formats without performing these steps separately.

Version 3.2 of SELECT uses the FORMAT command to format the disk in *drive2*, then uses the XCOPY command to copy all files on the disk in *drive1* (including the hidden system files) to *drive2*. If a country configuration other than one of the six KEYB*xx* utilities supplied on the distribution disk is specified, SELECT prompts the user to insert the disk containing the appropriate file.

Versions 3.0 and 3.1 of SELECT use the DISKCOPY program to copy all files on the disk in drive A (including the hidden system files) to the disk in drive B, formatting the disk if necessary.

All versions then add the appropriate CONFIG.SYS and AUTOEXEC.BAT files to the new disk to configure PC-DOS for use with the specified keyboard and country configuration. The specified configuration does not take effect until the computer is turned on or restarted using the new disk.

## Examples

To create a PC-DOS system disk configured for West Germany using version 3.0 or 3.1, place a copy of the original PC-DOS distribution disk in drive A and a blank disk in drive B; then type

```
A>SELECT 049 GR  <Enter>
```

During the copy operation, the usual DISKCOPY prompts and messages are displayed. When the copy operation is complete, the two disks are compared using DISKCOMP, producing the usual DISKCOMP prompts and messages. The resulting disk includes all the files from the distribution disk (including the hidden system files), a CONFIG.SYS file that contains the line

```
COUNTRY=049
```

and an AUTOEXEC.BAT file that contains the following lines:

```
KEYBGR
ECHO OFF
CLS
DATE
TIME
VER
```

To create a PC-DOS system disk configured for West Germany using version 3.2, place a copy of the original PC-DOS distribution disk in drive A and a blank disk in drive B; then type

```
A>SELECT 049 GR  <Enter>
```

SELECT first uses the FORMAT command to format the disk in drive B, then uses XCOPY to copy all files on the distribution disk (including the system files), and finally creates a CONFIG.SYS file that contains the line

```
COUNTRY=049
```

and an AUTOEXEC.BAT file that contains the following lines:

```
PATH \;
KEYBGR
ECHO OFF
CLS
DATE
TIME
VER
```

## Messages

**Cannot execute *X:filename***
One of the files needed by SELECT (FORMAT, DISKCOPY, DISKCOMP, or XCOPY) is not on the source disk or is a version that is not compatible with the version of PC-DOS that is running.

**File creation error**
The root directory of the destination disk is full or unable to contain any more files or one of the files being created has the same name as a directory already on the destination disk.

**Incorrect DOS version**
The version of SELECT is not compatible with the version of PC-DOS that is running (version 3.2).

**Incorrect number of parameters**
Too many or too few parameters were specified in the command line or a separator character was omitted between two parameters (version 3.2).

**Insert DOS diskette in drive A:**
**Strike any key when ready**
This message prompts the user to insert the distribution disk containing the system files and COMMAND.COM into drive A (version 3.2).

**Insert KEYB*xx*.COM diskette in drive *X*:**
**Strike any key when ready**
The user responded *Y* to a previous prompt asking if KEYB*xx* is on another disk. This message prompts the user to insert that disk into the specified drive (version 3.2).

**Insert target diskette in drive A:**
**Strike any key when ready**
This message prompts the user to insert the disk that will become the country-specific system disk into drive A (versions 3.0 and 3.1).

**Insert target diskette in drive B:**
**Strike any key when ready**
This message prompts the user to insert the disk that will become the country-specific system disk into drive B (version 3.2).

**Invalid country code**
The country code given in the command line is not supported by this version of PC-DOS or is not a valid country code.

**Invalid drive specification**
One of the drives specified in the command line is invalid or does not exist in the system (version 3.2).

**Invalid keyboard code**
The keyboard code given in the command line is not supported by this version of PC-DOS or is not a valid keyboard code.

**Invalid parameter**
One of the parameters specified in the command line is invalid or is not supported by the version of SELECT that is running (version 3.2).

**Invalid path**
The path specified for *drive2* is invalid, contains invalid characters, or is longer than 63 characters (version 3.2).

**Is KEYB*xx*.COM on another diskette (Y/N)?**
The keyboard reconfiguration file for the specified country is not on the source disk. Respond with *Y* to cause SELECT to prompt for the disk containing the keyboard file after the FORMAT operation is completed; respond with *N* to terminate the SELECT command (version 3.2).

**Keyboard routine not found.**
The user responded *N* to a previous prompt asking if KEYB*xx* is on another disk (version 3.2).

**SELECT is used to install DOS the first time. Select erases everything on the specified target and then installs DOS.**
**Do you want to continue (Y/N)?**
This message warns the user that the specified disk will be formatted and all files on the source disk will be copied over. Respond with *Y* to continue; respond with *N* to terminate the SELECT command (version 3.2).

**Unable to copy keyboard routine**
An error occurred while the KEYB*xx*.COM program was being copied. Use the CHKDSK command to check the keyboard program on the source disk for damage (version 3.2).

**Unable to create directory**
The directory specified in the command line was not created because a directory with the same name already exists on the destination disk, the root directory of the destination disk is full, one of the directory names specified in the path does not exist, or a file with the same name already exists (version 3.2).

# SET

2.0 and later

Set Environment Variable

Internal

## Purpose

Defines an environment variable and a string that is its value.

## Syntax

SET [*name=value*]

or

SET *name=*

where:

| | |
|---|---|
| *name* | is a string of characters that defines an environment variable; lowercase letters are automatically converted to uppercase. |
| *value* | is a string of characters, a pathname, or a filename that defines the current value of *name*; no case conversion is made for *value*. |

## Description

The environment is a series of null-terminated ASCII (ASCIIZ) strings that contains environment variables and their values. (An environment variable associates a string consisting of a filename, a pathname, or other literal data with a symbolic name that can be referenced by programs. The form of the association is *name=value*.) The original, or master, environment belongs to the command processor and is established when the system is turned on or restarted. When a program is subsequently executed by the command processor or by another program, the new program inherits a private copy of its parent's environment.

The SET command enables the user to add, change, or delete an environment variable from the command processor's environment. If *value* is not included in the SET command, MS-DOS deletes the environment variable *name* from the environment. If the SET command is issued with no parameters, MS-DOS displays the values of all the variables in the environment.

With MS-DOS versions 2.x and 3.x, two particular variables are always found in an environment: PATH and COMSPEC. These variables are initialized during the system startup process and tell COMMAND.COM which subdirectories to search for executable files and where to find the transient portion of COMMAND.COM for reloading (versions 3.0 and later). (By default, PATH is a null string and therefore searches only the current or specified directory.) These special environment variables are influenced by the PATH and SHELL commands, respectively, but can also be changed with SET commands. Note, however, that changing the value of COMSPEC with SET will serve no useful purpose—changing to a different command processor must be done using an appropriate SHELL

command in the CONFIG.SYS file (the system must be restarted for it to take effect). Note also that it is not necessary to use the SET command with the PATH or PROMPT commands — MS-DOS will automatically add their new values to the environment if they are changed.

The environment, which can be as large as 32 KB, can be an effective source of global configuration information to executing programs. For instance, the Microsoft C Compiler and Microsoft Object Linker use environment variables to locate *include* and object library files. Environment variables can also be referenced as replaceable parameters in batch files, using the form %*name*%.

Under normal circumstances, MS-DOS expands the environment as necessary when SET commands are entered. However, when a batch file is being interpreted or when terminate-and-stay-resident (TSR) utilities have been loaded, the size of the command processor's environment becomes fixed. Under these circumstances, a SET command may result in the error message *Out of environment space.*

With version 3.2, the initial size of the environment can be increased either by using the COMMAND command with the /P and /E:*nnnn* switches at the system prompt or by including a SHELL command specifying COMMAND.COM followed by the /E:*nnnn* switch in the CONFIG.SYS file. *See* USER COMMANDS: COMMAND; CONFIG.SYS: SHELL.

## Examples

To define the environment variable *USER* and set its value to *FRED*, type

```
C>SET USER=FRED  <Enter>
```

To change the value of the environment variable *USER* to *SALLY*, type

```
C>SET USER=SALLY  <Enter>
```

To delete the environment variable *USER* and its value from the environment, type

```
C>SET USER=  <Enter>
```

To display all the environment variables, type

```
C>SET  <Enter>
```

The output of this command will be in the following form:

```
COMSPEC=C:\DOS3\COMMAND.COM
PROMPT=$p$_$n$g
PATH=D:\BIN;C:\DOS3;C:\WP\WORD;C:\ASM;C:\MSC\BIN
INCLUDE=c:\msc\include;c:\windows\lib
LIB=c:\msc\lib;c:\windows\lib
TMP=c:\temp
PCF32=c:\forth\pc32
PROCOMM=c:\procomm\
```

## Message

**Out of environment space**
The command processor's environment is full and cannot be expanded (usually because the SET command was issued from a batch file or the system has terminate-and-stay-resident [TSR] utilities installed).

# SHARE

Install File-Sharing Support

3.0 and later

External

## Purpose

Loads the resident file-sharing support module required by Microsoft Networks.

## Syntax

SHARE [/F:*n*] [/L:*n*]

where:

| | |
|---|---|
| /F:*n* | allocates *n* bytes of memory to hold file-sharing information (default = 2048). |
| /L:*n* | configures support for *n* simultaneous file-region locks (default = 20). |

## Description

The code that supports file sharing and locking in a networking environment is isolated in the user-installable SHARE module. After SHARE is loaded, MS-DOS checks all read and write requests against the file-sharing module. On personal computers that do not utilize network services, the SHARE module need not be loaded, leaving more memory for application programs.

The /F:*n* switch controls the amount of buffer space allocated for file-sharing information. Each open file requires the length of its full name, including the path, plus some overhead; the average pathname is approximately 20 bytes long. If the /F:*n* switch is not included in the command line, the buffer size defaults to 2048 bytes (sufficient for approximately 100 files with pathnames of average length).

The /L:*n* switch controls the number of entries to be allocated for an internal table containing file-locking information. Each active lock on a region of a file occupies one entry in the table. If the /L:*n* switch is absent, the default is support for 20 simultaneously active locks.

## Example

To install the file-sharing support module, allocating 4096 bytes of space for file-sharing information and 40 file-region locks, type

```
C>SHARE /F:4096 /L:40  <Enter>
```

## Messages

**Incorrect DOS version**
The version of SHARE is not compatible with the version of MS-DOS that is running.

**Incorrect parameter**
The command line included an invalid switch.

**Not enough memory**
System memory is insufficient to load the SHARE module or to reserve the designated file-sharing information space or file-region locks.

**SHARE already installed**
The SHARE command has already been executed since the system was turned on or restarted; additional executions have no effect.

# SORT

Alphabetic Sort Filter

2.0 and later

External

## Purpose

Reads records from standard input, sorts them alphabetically, and writes the sorted records to standard output.

## Syntax

SORT [/R][/+*column*]

where:

| | |
|---|---|
| /R | specifies a reverse, or descending, alphabetic sort. |
| /+*column* | specifies the first column to be used for sorting each line (default = 1). |

## Description

The SORT program is a filter that reads lines from standard input until an end-of-file marker is reached, sorts the lines into alphabetic order, and writes the sorted lines to standard output.

Standard input defaults to the keyboard; standard output defaults to the video display. Because standard input can be redirected, the SORT filter can also accept input from another character device, a file, or the piped output of another program or filter. (The most common use of SORT is to sort the redirected input from an ASCII text file.) Similarly, the output of SORT can be redirected to any character device or file or can be piped to another program.

SORT normally orders the lines of the input text stream alphabetically using the entire line, starting with column 1 as the sort key. Tab characters are not expanded to spaces. If the character in the sort-key column of one line is identical with the character in the sort-key column of the next line, SORT checks the next column to the right to determine which line will go before the other. If the second columns are also identical, the search continues to the right until a differing column is found. The maximum amount of data that can be sorted is 63 KB.

The /R switch causes SORT to arrange the set of lines in reverse alphabetic order. The /+*column* switch lets the user specify a column other than column 1 as the first sort key.

With versions 2.x, SORT arranges the input lines based on the ASCII value of the character in each line's sort-key column; the sort operation is therefore case sensitive. With versions 3.0 and later, SORT assigns lowercase letters the same ASCII value as uppercase letters; hence, case is effectively ignored. Depending on the COUNTRY command in effect (*see* USER COMMANDS: CONFIG.SYS: COUNTRY), versions 3.0 and later map accented characters with ASCII codes in the range 80H through 0E1H (128–225) to their unaccented equivalents for sorting.

***Warning:*** If the output of the SORT command is redirected to a file with the same name as the input file, the contents of the input file may be destroyed.

## Examples

The examples in this entry operate on an ASCII text file named RECORDS.TXT that contains the following lines:

```
Smith     Seattle
Adams     New York
Zoole     Bellevue
Jones     Boston
```

Each line of the file contains a person's surname, starting in column 1, and a city name, starting in column 10.

To sort the file RECORDS.TXT by surname and display the sorted lines on standard output, type

```
C>SORT < RECORDS.TXT  <Enter>
```

This will result in the following display:

```
Adams     New York
Jones     Boston
Smith     Seattle
Zoole     Bellevue
```

To sort the file RECORDS.TXT by surname and write the sorted lines into the file READY.DOC, type

```
C>SORT < RECORDS.TXT > READY.DOC  <Enter>
```

To sort the file RECORDS.TXT by surname in reverse alphabetic order and display the sorted lines on standard output, type

```
C>SORT /R < RECORDS.TXT  <Enter>
```

This will result in the following display:

```
Zoole     Bellevue
Smith     Seattle
Jones     Boston
Adams     New York
```

To sort the file RECORDS.TXT by city name and display the sorted lines on standard output, type

```
C>SORT /+10 < RECORDS.TXT  <Enter>
```

This will result in the following display:

```
Zoole     Bellevue
Jones     Boston
Adams     New York
Smith     Seattle
```

To use SORT as a filter to arrange a directory listing alphabetically, type

```
C>DIR ¦ SORT  <Enter>
```

To use SORT as a filter to arrange a directory listing alphabetically based on the first character of each file's extension, type

```
C>DIR ¦ SORT /+10  <Enter>
```

## Messages

**Invalid parameter**
One of the parameters specified in the command line is invalid or the syntax is incorrect.

**SORT: Incorrect DOS version**
The version of SORT is not compatible with the version of MS-DOS that is running.

**SORT: Insufficient disk space**
The output of the SORT filter has been redirected to a file and the disk is full.

**SORT: Insufficient memory**
The available system memory is insufficient to run the SORT program.

# SUBST

Substitute Drive for Subdirectory

3.1 and later

External No Net

## Purpose

Causes a drive letter to be substituted for a directory name. SUBST is present in MS-DOS to support older application programs that do not accept pathnames.

## Syntax

SUBST [*drive1*: [*drive2*:]*path*]

or

SUBST *drive1*: /D

where:

| | |
|---|---|
| *drive1* | is the drive letter to be used to reference the files in *path*. |
| *drive2* | is a drive letter other than *drive1* that can optionally precede the name of the subdirectory being substituted. |
| *path* | is the subdirectory to be accessed when *drive1* is referenced, optionally preceded by *drive2*. |
| /D | cancels the effect of a previous SUBST command for *drive1*. |

## Description

The SUBST command allows a drive letter to be substituted for a subdirectory name.

The *drive1* parameter can be any valid drive letter except the current drive or *drive2*. Drive letters A through E are always available; drive letters beyond E require that an appropriate LASTDRIVE command be added to the CONFIG.SYS file and the system be restarted (*see* USER COMMANDS: CONFIG.SYS: LASTDRIVE).

After a SUBST command, the files on the disk normally referenced by *drive1* are no longer accessible. However, the files in the location specified by *path* can still be referenced by the usual methods (using their actual drive and path) as well as by the substituted drive designator.

If the SUBST command is entered without parameters, MS-DOS displays the substitutions currently in effect.

***Warning:*** The SUBST command masks the actual disk-drive characteristics from commands that perform critical disk operations. Therefore, ASSIGN, BACKUP, CHKDSK, DISKCOMP, DISKCOPY, FDISK, FORMAT, JOIN, LABEL, and RESTORE should not be used on a drive affected by a SUBST command. CHDIR, MKDIR, RMDIR, and PATH commands that include the affected drive should be used with caution. A network drive cannot be named in a SUBST command.

## Examples

To substitute drive B for the directory C:\ASM\SOURCE, type

```
C>SUBST B: C:\ASM\SOURCE  <Enter>
```

To display the substitutions currently in effect, type

```
C>SUBST  <Enter>
```

In this case, the SUBST command displays

```
B: => C:\ASM\SOURCE
```

To cancel the effect of a previous SUBST command that substituted drive B for a subdirectory, type

```
C>SUBST B: /D  <Enter>
```

## Messages

**Cannot SUBST a network drive**
One or both of the drive parameters in the command line referred to a drive that is assigned to a network.

**DOS 2.0 or later required**
SUBST does not work with versions of MS-DOS earlier than 2.0.

**Incorrect DOS version**
The version of SUBST is not compatible with the version of MS-DOS that is running.

**Incorrect number of parameters**
The command line included too many or too few parameters.

**Invalid parameter**
The drive named in the command line is invalid, does not exist, is the default drive, or is the same as the drive in the path to be substituted.

**Not enough memory**
The available system memory is insufficient to run the SUBST command.

**Path not found**
An element of the path included in the command line is invalid or does not exist.

# SYS

1.0 and later

Transfer System Files

External No Net

## Purpose

Copies the hidden files that contain the operating system from the disk in the current drive to another formatted disk.

## Syntax

SYS *drive*:

where:

*drive* is the location of the disk that will receive the system files. This parameter is required.

## Description

An MS-DOS system disk must contain three files to be bootable: the two operating-system files and the command processor. The operating system itself is contained in the files IO.SYS and MSDOS.SYS (or IBMBIO.COM and IBMDOS.COM in PC-DOS), which must always be the first two files in the disk's directory. Both have file attributes set for system and hidden (all versions) and read-only (versions 2.0 and later). IO.SYS (or IBMBIO.COM) contains the default set of device drivers for the system; it must occupy contiguous sectors in the disk's files area. MSDOS.SYS (or IBMDOS.COM) contains the kernel of the operating system proper. The third required file is the shell, or command processor, which by default is COMMAND.COM. This is an unrestricted file and can be located anywhere on the disk.

The SYS command transfers the two operating-system files from the default drive to the specified destination disk. The destination disk that receives the files must meet one of the following requirements:

- The disk is formatted but completely empty.
- The disk currently contains hidden MS-DOS system files that are large enough to allow replacement by the new system files.
- The disk has been formatted with the /B switch to reserve room for the system files. (Note that /B produces a disk with only eight sectors per track.)

If the disk already contains the two hidden system files, the SYS command can be used to transfer an equivalent or later version of MS-DOS.

After the two hidden operating-system files are installed with the SYS command, the COMMAND.COM file (or another command processor) must be transferred to the destination disk with the COPY command. The resulting disk is a bootable system disk.

***Note:*** Because the two system files have the hidden attribute, they do not appear on a directory listing produced by the DIR command. The CHKDSK command does report the presence of hidden files on a disk and will list their names if the /V switch is used but will not list such information as the file size or date and time of creation.

## Example

To transfer a copy of the system files to the disk in drive B, type

```
C>SYS B:  <Enter>
```

## Messages

**Cannot SYS to a Network drive**
The drive specified in the command line is currently assigned to a network.

**Destination disk cannot be booted**
The hidden operating-system files were transferred to the destination disk but could not be placed in contiguous sectors.

**Incompatible system size**
The destination disk already contains operating-system files and they are smaller than those being copied.

**Incorrect DOS version**
The version of SYS is not compatible with the version of MS-DOS that is running.

**Insert destination disk in drive *X***
**and strike any key when ready**
This message prompts the user to insert the disk onto which the operating-system files will be copied into the specified drive.

**Insert system disk in drive *X***
**and strike any key when ready**
This message prompts the user to insert a disk containing the operating-system files into the specified drive.

**Invalid drive specification**
The drive specified in the command line is invalid or does not exist in the system.

**Invalid parameter**
The command line contained an invalid drive letter.

**No room for system on destination disk**
Contiguous space at the beginning of the destination disk is insufficient for the operating-system files. This can occur when files already exist on the destination disk or when sections of the disk are marked as unusable by the FORMAT command.

**No system on default drive**
The disk in the default drive does not contain the two hidden system files. Replace the disk with a bootable system disk.

**System transferred**
The operating-system files have been successfully transferred to the destination disk.

# TIME

1.0 and later

Set System Time

Internal

## Purpose

Sets or displays the system time. TIME is an external command with PC-DOS version 1.0.

## Syntax

TIME [*hh*:*mm*[:*ss*[.*xx*]]]

where:

| | |
|---|---|
| *hh* | is hours (0-23). |
| *mm* | is minutes (0-59). |
| *ss* | is seconds (0-59). |
| *xx* | is hundredths of a second (0–99). |

***Note:*** No spaces are allowed between any of the time parameters.

## Description

All computers that run MS-DOS have as part of their hardware configuration a timer, or clock, that maintains the current system date and time. One use of this clock, among others, is to insert the current date and time into a file's directory entry when the file is created or modified.

The TIME command allows the user to display or modify the current time that is being maintained by the system's real-time clock. TIME is also executed by MS-DOS when the system is turned on or restarted, unless an AUTOEXEC.BAT file is on the system disk, in which case the command is executed only if it is included in the AUTOEXEC.BAT file.

On IBM PC/ATs and compatibles, the TIME command does not permanently change the system time stored in the built-in battery-backed clock/calendar; the newly entered time is lost when the system is turned off or restarted. On these machines, the SETUP program (found on the *Diagnostics for IBM Personal Computer AT* disk or equivalent) must be used to permanently alter the clock/calendar's current time.

On IBM PCs, PC/XTs, and compatibles equipped with add-on cards containing battery-backed clock/calendar circuitry, it is usually necessary to run a time/date installation program (included with the card) to set the system date and time from the clock/calendar on the card. The TIME command generally has no effect on these card-mounted clock/calendars.

The format of times displayed by the system depends on the current country code, which is determined by the optional COUNTRY command in the CONFIG.SYS file (*see* USER COMMANDS: CONFIG.SYS: COUNTRY). The default display format is the 24-hour format (00:00–23:59).

## Examples

To display the current time, type

```
C>TIME  <Enter>
```

This results in output of the following form:

```
Current time is 12:49:04.93
Enter new time:
```

To leave the time unchanged, press the Enter key.

To set the system time to 8:30 P.M., type

```
C>TIME 20:30  <Enter>
```

## Messages

**Current time is *hh:mm:ss.xx***
This informational message is displayed in response to any valid TIME command.

**Invalid parameter**
The delimiter in the time parameter included in the command line was not a colon (:) or a period (.).

**Invalid time**
**Enter new time:**
An invalid time, time format, or delimiter was specified in the command line or in response to the *Enter new time:* prompt. Note that no spaces are allowed around delimiters.

# TREE

3.2

Display Directory Structure

External

## Purpose

Displays the hierarchical directory structure of a disk and, optionally, the names of the files in each subdirectory. This command is included with PC-DOS beginning with version 2.0.

## Syntax

TREE [*drive*:][/F]

where:

| | |
|---|---|
| *drive* | is the location of the disk whose directory structure is to be displayed. |
| /F | displays the filenames in each directory in addition to the directory names. |

## Description

The TREE command displays on standard output the pathname of each directory on the disk in the specified drive, beginning with the subdirectories of the root directory. If a disk drive is not designated, TREE assumes the current, or default, drive. The name of each directory is followed by a list of its subdirectories. If the /F switch is included in the command line, the names of the files in each subdirectory are also displayed. (Prior to version 3.1, the PC-DOS TREE command does not list the files in the root directory if /F is used.)

The output of the TREE command can be redirected to another output device or a file or can be piped to another program.

## Examples

Assume that the root directory of the disk in drive B contains three subdirectories: \SOURCE, \LIBS, and \DOC. The subdirectory \SOURCE in turn contains two subdirectories: \ASM and \PASCAL. To display the directory structure of this disk, type

```
C>TREE B:  <Enter>
```

The TREE command displays the following list:

```
DIRECTORY PATH LISTING FOR VOLUME MYDISK

Path: B:\SOURCE

Sub-directories:  ASM
                  PASCAL


Path: B:\SOURCE\ASM

Sub-directories:  None


Path: B:\SOURCE\PASCAL

Sub-directories:  None


Path: B:\LIBS

Sub-directories:  None


Path: B:\DOC

Sub-directories:  None
```

To display the directory structure of the disk in drive B and also display all files in each directory, type

```
C>TREE B: /F  <Enter>
```

To print the directory-structure listing of the disk in drive B on an attached printer, type

```
C>TREE B: > PRN  <Enter>
```

To display the directory structure of the disk in drive B one screenful at a time, type

```
C>TREE B: ¦ MORE  <Enter>
```

For a more compressed listing of all subdirectories on the disk in drive B, type

```
C>TREE B: ¦ FIND "Path:"  <Enter>
```

The output appears in the following form:

```
Path: B:\SOURCE
Path: B:\SOURCE\ASM
Path: B:\SOURCE\PASCAL
Path: B:\LIBS
Path: B:\DOC
```

## Messages

**DOS 2.0 or later required**
TREE does not work with versions of MS-DOS earlier than 2.0.

**Incorrect DOS version**
The version of TREE is not compatible with the version of MS-DOS that is running.

**Invalid drive specification**
The drive specified in the command line is invalid or does not exist in the system.

**Invalid parameter**
The command line contained a path or filename in addition to a disk drive or contained an invalid switch.

**No sub-directories exist**
The specified drive has no subdirectories.

# TYPE

Display File

1.0 and later

Internal

## Purpose

Sends the contents of an ASCII text file to standard output.

## Syntax

TYPE [*drive*:][*path*]*filename*

where:

*filename* is the name of the text file to be displayed, optionally preceded by a drive and/or path; wildcard characters are not permitted.

## Description

The TYPE command displays the contents of a text file on standard output (usually the video display) until it encounters an end-of-file character (ASCII code 1AH). Tab characters in the file are expanded to spaces with tab stops at each eighth character position. If a file contains characters with ASCII values less than 32 or greater than 127, the resulting display includes graphics characters and other unintelligible information.

The output of the TYPE command can be redirected to another file or character device or can be piped to another program.

## Examples

To display the file SHELL.C in the directory \SOURCE on the disk in drive A, type

```
C>TYPE A:\SOURCE\SHELL.C  <Enter>
```

To direct the output of the same file to the printer, type

```
C>TYPE A:\SOURCE\SHELL.C > PRN  <Enter>
```

The TYPE command can be used with the MORE filter to paginate output. For example, to display the contents of the file MENU.ASM one screenful at a time, type

```
C>TYPE MENU.ASM | MORE  <Enter>
```

## Messages

**File not found**
The file specified in the command line cannot be found or does not exist.

**Invalid drive specification**
The drive specified in the command line is invalid or does not exist in the system.

**Invalid path or file name**
The path specified in the command line is invalid or does not exist.

# VDISK.SYS

IBM

Virtual Disk

External

## Purpose

Creates a virtual disk in memory. This installable driver is available only with PC-DOS.

## Syntax

DEVICE=[*drive*:][*path*]VDISK.SYS [*size*] [*sector*] [*directory*] [/E] (version 3.0)

or

DEVICE=[*drive*:][*path*]VDISK.SYS [*size*] [*sector*] [*directory*] [/E[:*max*]] (version 3.1)

or

DEVICE=[*drive*:][*path*]VDISK.SYS [*comment*] [*size*] [*comment*] [*sector*] [*comment*] [*directory*] [/E[:*max*]] (version 3.2)

where:

| | |
|---|---|
| *comment* | is a string of ASCII characters in the range 32 through 126, excluding the slash character (/) (version 3.2). |
| *size* | is the size of the virtual disk in kilobytes (minimum = 1, default = 64). |
| *sector* | is the sector size in bytes (128, 256, or 512; default = 128). |
| *directory* | is the maximum number of entries in the virtual disk's root directory (2–512, default = 64). |
| /E | causes VDISK to use extended memory. |
| /E:*max* | causes VDISK to use extended memory and sets the maximum number of sectors (1–8, default = 8) to transfer from extended memory at one time (versions 3.1 and later). |

***Note:*** Unless the /E switch is used, the virtual disk is created in conventional memory.

## Description

The VDISK.SYS installable device driver allows the configuration of one or more virtual disks (sometimes referred to as electronic disks or RAMdisks). A virtual disk is implemented by mapping a disk's structure — directory, file allocation table, and files area — onto an area of random-access memory, rather than onto actual sectors located on a magnetic recording medium. Access to files stored in a virtual disk is very fast, because no moving parts are involved and the "disk" operates at the speed of the system's memory. (The VDISK driver is available only with PC-DOS; a similar program named RAMDRIVE.SYS is included with MS-DOS.)

***Warning:*** Because a RAMdisk resides entirely in RAM and is therefore volatile, any information stored there is irretrievably lost when the computer loses power or is restarted.

VDISK can create a virtual disk in either conventional memory or extended memory. Conventional memory is the term for the up-to-640 KB of RAM that contain PC-DOS and any application programs. Extended memory is the term for the memory at addresses above 1 MB (100000H) that is available on 80286-based personal computers such as the IBM PC/AT.

A virtual disk can be installed in conventional memory by simply inserting the line *DEVICE=VDISK.SYS* into the system's CONFIG.SYS file and restarting the system. (If the file VDISK.SYS is not in the root directory of the startup disk, it may be preceded by a drive and/or path.) A new "drive" then becomes available in the system, with default values of 64 KB disk size, 128-byte sectors, and 64 available directory entries (assuming there is sufficient memory). The virtual disk is assigned the next available drive letter (which is displayed in VDISK's sign-on message). The drive letter assigned depends on the number of other physical and virtual disks in the system and also on the position of the *DEVICE=VDISK.SYS* line in the CONFIG.SYS file relative to other installed block devices. Available memory permitting, multiple virtual disks can be created by using multiple *DEVICE=VDISK.SYS* lines. Several optional parameters allow the user to customize the size and configuration of the virtual disk and to use extended memory if it is available.

The *size* parameter specifies the amount of RAM, in kilobytes, to be allocated to the virtual disk. The default is 64 KB, but any size from 1 KB to the total amount of available memory can be specified. If the size specified is greater than available memory or less than 1 KB, VDISK ignores it and creates a virtual disk of 64 KB. If necessary, VDISK also adjusts the *size* value to ensure that at least 64 KB of memory remain available in the system.

The *sector* parameter sets the virtual sector size used within the virtual disk. The *sector* value may be 128, 256, or 512 bytes (default = 128 bytes). Selection of the smallest sector size results in a minimum of wasted virtual disk space per file but also results in somewhat slower transfer of data.

***Note:*** Physical disk devices in IBM PC-compatible systems always use 512-byte sectors.

The *directory* parameter sets the number of available entries in the virtual disk's root directory. The allowed range is 2 through 512 (default = 64). Each directory entry requires 32 bytes. VDISK rounds the number of available directory entries up, if necessary, so that an integral number of sectors are assigned to the root directory.

The /E switch causes VDISK to use extended memory for the virtual disk, rather than conventional memory. This allows very large virtual disks to be configured while still leaving the maximum amount of conventional memory available for use by application programs. If the /E switch is used and extended memory is not present in the system, the VDISK driver will not install itself.

When /E is used in the form /E:*max*, the variable *max* controls how many virtual sectors can be transferred at a time from extended memory. The value of *max* must be in the range 1 through 8 (default = 8). If VDISK operation appears to conflict with the communications port or other interrupt-driven peripheral devices, the *max* variable should be set to a smaller number. The *max* option is available only with versions 3.1 and 3.2.

***Note:*** If VDISK uses conventional memory for virtual disk storage, the memory cannot be reclaimed except by modifying the CONFIG.SYS file and restarting the system.

## Examples

To create a virtual disk drive with the default values of 64 KB disk size, 128-byte sectors, and 64 available directory entries, include the command

```
DEVICE=VDISK.SYS
```

in the CONFIG.SYS file and restart the system.

To create a 360 KB virtual disk with 512-byte sectors and 112 available directory entries when the file VDISK.SYS is located in a directory named \BIN on drive C, include the command

```
DEVICE=C:\BIN\VDISK.SYS 360 512 112
```

in the CONFIG.SYS file and restart the system. The directory for this virtual disk requires 3584 bytes (112 entries * 32 bytes), or 7 sectors.

With version 3.2, comments can be inserted between the values to identify them. For example, to create a 1 MB virtual disk drive in extended memory with 256-byte sectors and 128 directory entries, placing comments before the values to identify them, include the command

```
DEVICE=VDISK.SYS DISK_SIZE: 1024 SECTOR_SIZE: 256 DIR_ENTRIES: 128 /E
```

in the CONFIG.SYS file and restart the system.

## Messages

**Buffer size adjusted**
No *size* value was specified or the specified value was larger than the amount of available memory.

**Directory entries adjusted**
No *directory* value was specified, VDISK adjusted the *directory* value up to the nearest sector-size boundary, or the *size* value was too small to hold the file allocation table, the directory, and two additional sectors, in which case VDISK adjusted *directory* downward until these conditions were met.

**Invalid switch character**
A slash character (/) was included in a comment or the /E switch was entered incorrectly.

**Sector size adjusted**
The *sector* value was missing from the command line or an incorrect value was entered; therefore, VDISK used the default value of 128 bytes.

**Transfer size adjusted**
A value outside the range 1 through 8 was specified with the /E:*max* switch; therefore, VDISK used the default value of 8.

**VDISK not installed - Extender Card switches do not match the system memory size**
The switch settings on the extender card are not correct or the extended memory exists in an expansion unit, which VDISK is not capable of using.

**VDISK not installed - insufficient memory**
Less than 64 KB of system memory remained after attempted installation, the /E switch was specified and the system does not contain extended memory, or the amount of available extended memory was too small to support the installation of VDISK.

**VDISK Version *n.nn* virtual disk *X*:**
**Buffer size: *nn* KB**
**Sector size: *nnn***
**Directory size: *nnn***
**Transfer size: *n***
VDISK was successfully installed and this message informs the user of the drive letter assigned to the virtual disk, the version of VDISK that created the disk, and the characteristics of the disk. The *Transfer size:* message appears only in versions 3.1 and 3.2 and only if the /E switch was used.

# VER

2.0 and later

Display Version

Internal

## Purpose

Displays the MS-DOS version number.

## Syntax

VER

## Description

The VER command displays on standard output (usually the video display) the number of the MS-DOS version that is running. The version number is also displayed as part of the copyright notice when the system is turned on or restarted, unless an AUTOEXEC.BAT file is on the system disk. (The VER command can be included in the AUTOEXEC.BAT file to display the version number, but it will not display the copyright information.)

## Examples

To display the MS-DOS version number, type

```
C>VER  <Enter>
```

On a system that is running MS-DOS version 3.2, the following message is displayed:

```
MS-DOS Version 3.2
```

To print the MS-DOS version number on an attached printer instead of displaying it on the screen, type

```
C>VER > PRN  <Enter>
```

# VERIFY

Set Verify Flag

2.0 and later

Internal

## Purpose

Sets the system's internal flag controlling verification of disk writes.

## Syntax

VERIFY [ON|OFF]

## Description

The VERIFY command sets or clears an internal MS-DOS flag that controls verification of data written to disks. (The actual verification process is usually carried out by the device driver and the disk-drive controller.) The VERIFY ON command has the same effect on a global basis as the /V switch has on COPY operations. (When VERIFY is on, use of the /V switch with COPY has no additional effect.) VERIFY ON remains in effect until a program turns it off with a Set Verify system call or until the user types *VERIFY OFF* at the command prompt. The VERIFY command does not affect the operation of character devices.

When the VERIFY command is entered without an ON or OFF, MS-DOS displays the current state of the system's internal verify flag. The default setting of the verify flag is off.

## Examples

To turn on verification of disk writes, type

```
C>VERIFY ON  <Enter>
```

To display the current status of the verify flag, type

```
C>VERIFY  <Enter>
```

## Messages

**Must specify ON or OFF**
The command line contained an invalid parameter.

**VERIFY is off**

or

**VERIFY is on**
No setting was specified in the command line and VERIFY displays this informational message indicating the current status of the verify flag.

# VOL

2.0 and later

Display Disk Name

Internal

## Purpose

Displays a disk's volume label if one exists.

## Syntax

VOL [*drive*:]

where:

*drive* is the location of the disk whose volume label is to be displayed.

## Description

The VOL command displays a disk's name, or volume label. If *drive* is not included in the command line, the volume label of the disk in the current drive is displayed.

A volume label can be assigned to a disk when it is formatted by using the /V switch with the FORMAT command. A volume label can be added, changed, or deleted *after* a disk has already been formatted by using the LABEL command (PC-DOS versions 3.0 and later, MS-DOS versions 3.1 and later). The CHKDSK, DIR, and TREE commands also display a disk's volume label as part of their output.

## Example

To display the volume label for the disk in the current drive, type

```
C>VOL  <Enter>
```

If the disk's name is HARDDISK, the VOL command produces the following output:

```
Volume in drive C is HARDDISK
```

## Messages

**Invalid drive specification**
The drive specified in the command line is invalid or does not exist in the system.

**Volume in drive *X* has no label**
The disk in the current or specified drive was not previously assigned a volume label with the FORMAT or LABEL command.

# XCOPY

Copy Files

3.2

External

## Purpose

Copies files and directories, optionally also copying subdirectories and the files they contain.

## Syntax

XCOPY *source* [*destination*][/A] [/D:*mm-dd-yy*] [/E] [/M] [/P] [/S] [/V] [/W]

where:

| | |
|---|---|
| *source* | is the name of the file(s) to be copied, optionally preceded by a drive and/or path; wildcard characters are permitted in the filename. If the path is omitted, a drive letter must be specified; this parameter is not optional. |
| *destination* | is the destination location and, optionally, the name for the copied files, and can be preceded by a drive; wildcard characters are permitted in the filename. |
| /A | copies only those source files with the archive bit set. |
| /D:*mm-dd-yy* | copies only files modified on or after the specified date. (The date format depends on the COUNTRY command in effect, if any.) |
| /E | copies empty subdirectories; if this switch is used, the /S switch must also be specified. |
| /M | copies only those files with the archive bit set; also turns off the archive bit of each source file after it is copied. |
| /P | prompts the user for confirmation before copying each file. |
| /S | copies all nonempty subdirectories of *source* and the files they contain. |
| /V | performs read-after-write verification of destination file(s). |
| /W | waits for the user to press a key before copying any files, allowing disks to be changed. |

## Description

The XCOPY command copies one or more source files to one or more destination files. Unlike the COPY command, however, a single XCOPY command can copy *all* files contained in the entire hierarchical file structure of the source disk to the destination disk, creating a corresponding set of directories and subdirectories at the destination to hold the copied files.

The *source* parameter identifies the file or files to be copied. It can consist of any combination of a drive, path, and filename (optionally including wildcards) but *must* include either

a drive or a pathname. If only a drive is specified, all files in the current directory of that drive are copied. If a path without a drive or filename is specified, all files in the named directory are copied from the current drive.

The *destination* parameter can also consist of any combination of drive, path, and filename. Unless only a single file is being copied and it is also being renamed as part of the XCOPY operation, *destination* is usually simply a drive and/or path specifying where to place the copied file. If *destination* includes a filename, XCOPY displays a message asking if the specified destination is a file or a directory. Depending on the user's response, XCOPY then either copies the source file to a destination file with the specified name or creates a directory with the specified name and copies the source files into it. (Note that if the user responds that the destination is to be a file and multiple source files were specified in the command line, only the last source file is copied to the specified destination.) If no destination is specified, the source file is copied to a file with the same name in the current directory of the current drive.

The /A, /D:*mm-dd-yy*, /M, and /P switches allow selective copying of files. The /A switch is used to copy only source files with the archive bit set; the /M switch also copies only source files with the archive bit set but turns off each source file's archive bit after the file is copied. The /D:*mm-dd-yy* switch is used to copy files that were modified on or after a selected date; the date must be entered in one of the formats discussed in the entry for the system's DATE command or in the format of the COUNTRY command currently in effect (*see* USER COMMANDS: CONFIG.SYS: COUNTRY). The /P switch causes XCOPY to prompt the user for confirmation before transferring each file.

The /E and /S switches allow an entire branch of the source disk's hierarchical directory structure to be copied. If the /S switch is specified, XCOPY copies all nonempty subdirectories of *source*, creating equivalent destination subdirectories, if necessary, to hold the files. If the /E switch is specified, XCOPY also duplicates empty source subdirectories in the equivalent destination locations. If the /E switch is used, the /S switch must also be specified.

The /V switch causes a Verify call to be issued on the destination file(s) to ensure that the data was written correctly. Its effect is equivalent to that of the VERIFY ON command.

Finally, the /W switch causes XCOPY to wait for the user to press a key before copying any files, thus allowing an exchange of disks before the files are transferred. This is useful in systems without a fixed disk, because it allows XCOPY to be used when the program itself is not on either the source or the destination disk.

***Note:*** With MS-DOS versions of XCOPY, the related program MCOPY can be created by simply copying the file XCOPY.EXE to a file named MCOPY.EXE using the following command:

```
C>COPY /B XCOPY.EXE MCOPY.EXE  <Enter>
```

What distinguishes MCOPY from XCOPY is the program name; when either program is loaded, it looks at the name under which it was invoked and reconfigures itself accordingly. MCOPY's behavior is similar to XCOPY's, except that MCOPY automatically

determines whether the name specified as the destination is a file or a directory according to the following rules:

- If the source is a directory, the specified destination is a directory.
- If the source includes multiple files, the specified destination is a directory.
- If the destination name ends with a backslash character (\), the specified destination is a directory.

MCOPY supports all the XCOPY switches.

Not all implementations of XCOPY can be renamed to MCOPY and function accordingly. The PC-DOS version of XCOPY, for example, does not support this feature.

## Return Codes

0 No errors were detected during the copy operation.
1 No files were found to copy.
2 The copy operation was terminated by a Ctrl-C or Ctrl-Break.
4 Initialization error occurred: not enough memory, file not found, or command-line syntax error.
5 The copy operation was terminated by an *A* response to an *Abort, Retry, Ignore?* prompt.

## Examples

To copy all files in the directory C:\SOURCE to the directory C:\SOURCE\BACKUP, type

```
C>XCOPY C:\SOURCE\*.* C:\SOURCE\BACKUP  <Enter>
```

To copy all files and directories on drive C to the disk in drive D, type

```
C>XCOPY C:\*.* D: /S /E  <Enter>
```

## Messages

***nn* File(s) copied**
This informational message is displayed at the completion of an XCOPY command and indicates the total number of source files processed.

***filename* File not found**
The source file specified in the command line is invalid or does not exist.

***X:pathname* (Y/N)?**
The /P switch was specified in the command line. XCOPY displays the name of each file, preceded by a drive (and path, if one was specified), and asks for confirmation before copying the file.

**Access denied**
A destination file could not be overwritten because it was marked read-only.

**Cannot COPY from a reserved device**
A character device such as AUX or COM1 cannot be the source of an XCOPY operation.

**Cannot COPY to a reserved device**
A character device such as PRN cannot be the destination of an XCOPY operation.

**Cannot perform a cyclic copy**
The command line included a /S switch and the destination directory is a subdirectory of the source directory. A subdirectory cannot be copied onto itself.

**Does *name* specify a file name or directory name on the target (F = file, D = directory)?**
The specified destination directory does not already exist; the user is prompted to determine whether it should be created. Respond with *F* to copy the source file to a file named *name*; respond with *D* to create a subdirectory named *name* and copy the source file into it.

**File cannot be copied onto itself**
The name and location of the source file are the same as the name and location of the destination file.

**File creation error**
A destination file or directory could not be created. The destination disk may be full.

**Incorrect DOS version**
The version of XCOPY is not compatible with the version of MS-DOS that is running.

**Insufficient disk space**
The disk does not contain enough available space to perform the specified XCOPY operation.

**Insufficient memory**
The available system memory is insufficient to perform the XCOPY operation.

**Invalid date**
The command included a /D switch and the date was not formatted properly.

**Invalid drive specification**
The source or destination drive specified in the command line is not valid or does not exist in the system.

**Invalid number of parameters**
The command line contained too many or too few filenames or other parameters.

**Invalid parameter**
A switch supplied in the command line is not valid.

**Invalid path**
A directory specified in the command line is invalid or does not exist.

**Lock Violation**
XCOPY attempted to access a file in use by another program. Respond with *A* to the error-message prompt and try XCOPY later or wait for a few minutes and respond with *R*.

**Path not found**
One of the pathnames specified in the command line is invalid or does not exist.

**Path too long**
The path element of the source or destination parameter was longer than 63 characters.

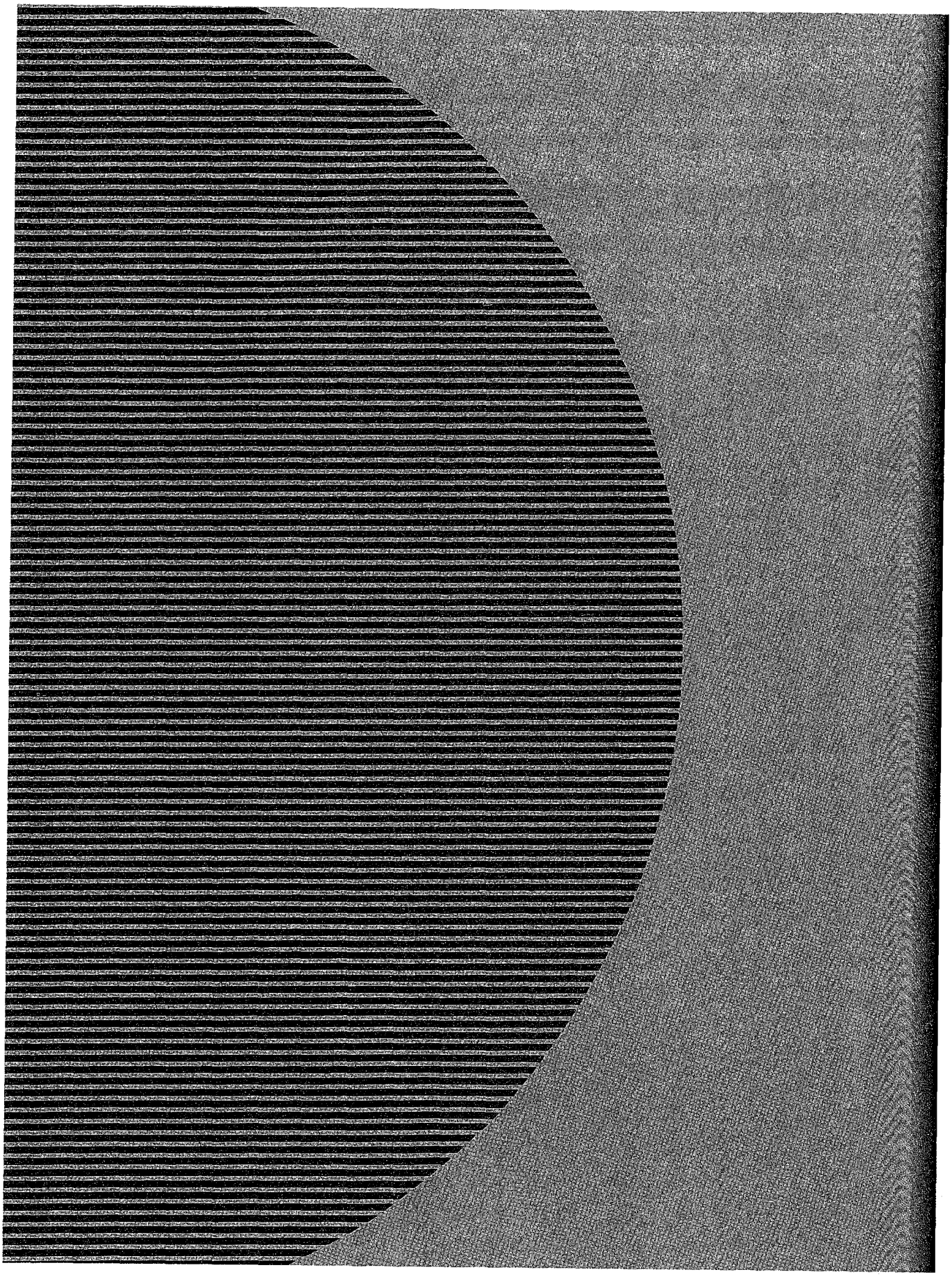
**Press any key to begin copying file(s)**
The /W switch was specified in the command line and XCOPY waits for the user to press a key before beginning the copy process.

**Reading source file(s)...**
This informational message is displayed during the XCOPY operation.

**Sharing violation**
XCOPY attempted to access a file in use by another program. Respond with *A* to the error-message prompt and try XCOPY later or wait a few minutes and respond with *R*.

**Too many open files**
XCOPY failed due to a lack of available system file handles. Increase the size of the FILES command in the CONFIG.SYS file, restart the system, and attempt the XCOPY command again.

**Unable to create directory**
A destination directory cannot have the same name as an existing file in the prospective parent directory.



# Section IV
# Programming Utilities

# Introduction

This section of *The MS-DOS Encyclopedia* describes the Microsoft utilities, documentation aids, and debuggers that can be used with the Microsoft C, FORTRAN, Pascal, and BASIC compilers and with the Microsoft Macro Assembler (MASM). Included are operating instructions for MASM, the Macro Assembler; LIB, the Library Manager; LINK, the Microsoft Object Linker; the DEBUG, SYMDEB, and CodeView program debuggers; MAKE, which automates maintenance of programs; CREF, which produces a cross-reference listing of symbols; and EXE2BIN, EXEMOD, and EXEPACK, which modify executable files.

Entries (except for the program debuggers) are arranged alphabetically by the name of the programming utility. The three Microsoft debuggers are listed at the end of the section in the following order: DEBUG, SYMDEB, CodeView. Individual DEBUG and SYMDEB commands appear alphabetically under the headings DEBUG and SYMDEB.

Each utility entry includes

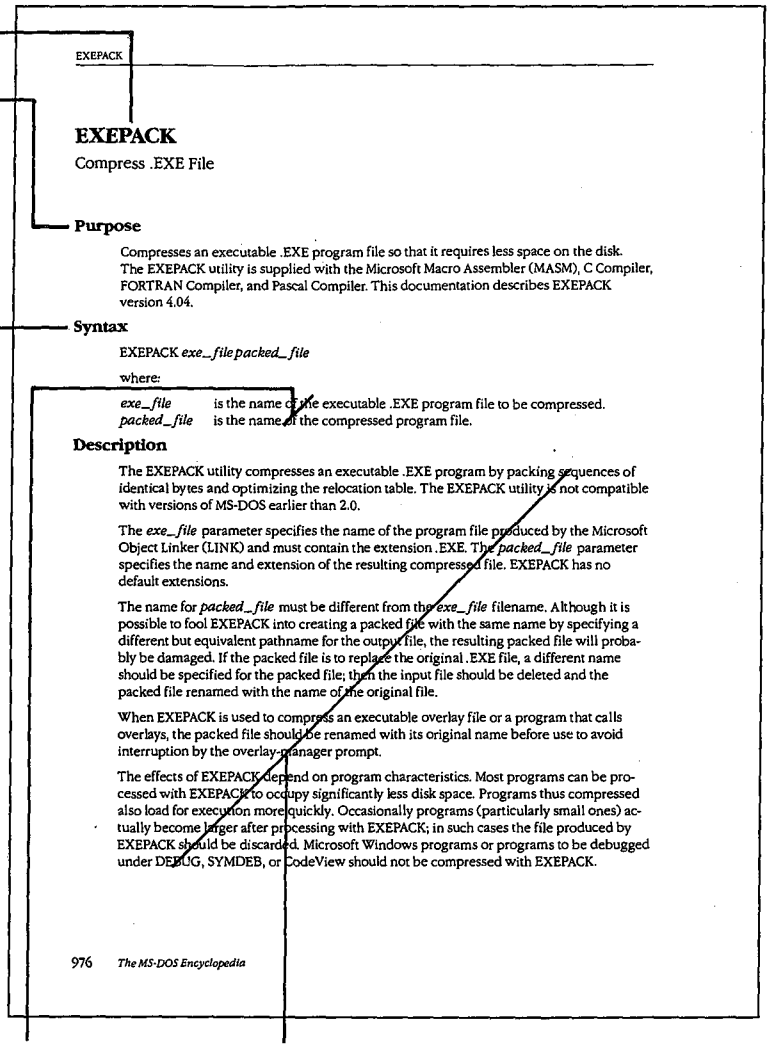
- Utility name
- Utility purpose
- Prototype command line and summary of options
- Detailed description of utility
- One or more examples of utility use
- Return codes (where applicable)
- Error messages and warnings (where applicable)

The experienced user can find information with a quick glance at the first part of a utility entry; a less experienced user can refer to the detailed explanation and examples in a more leisurely fashion. The next two pages contain an example of a typical entry from the Programming Utilities section, with explanations of each component.

HEADING
The utility name.

PURPOSE
An abstract of utility purpose and usage plus a statement of which Microsoft products the utility is supplied with and the utility version described in the entry.

SYNTAX
A prototype command line, with variable names in italic and optional parameters in square brackets. The various elements of the command line should be entered in the order shown. Any punctuation must be used exactly as shown; in commands that use commas as separators, the comma usually must be included as a placeholder even if the parameter is omitted. Except where noted, commands, parameters, and switches can be entered in either uppercase or lowercase. Utility names can be preceded by a drive and/or path.



EXEPACK

**EXEPACK**

Compress .EXE File

**Purpose**

Compresses an executable .EXE program file so that it requires less space on the disk. The EXEPACK utility is supplied with the Microsoft Macro Assembler (MASM), C Compiler, FORTRAN Compiler, and Pascal Compiler. This documentation describes EXEPACK version 4.04.

**Syntax**

EXEPACK *exe_file packed_file*

where:

*exe_file* is the name of the executable .EXE program file to be compressed.
*packed_file* is the name of the compressed program file.

**Description**

The EXEPACK utility compresses an executable .EXE program by packing sequences of identical bytes and optimizing the relocation table. The EXEPACK utility is not compatible with versions of MS-DOS earlier than 2.0.

The *exe_file* parameter specifies the name of the program file produced by the Microsoft Object Linker (LINK) and must contain the extension .EXE. The *packed_file* parameter specifies the name and extension of the resulting compressed file. EXEPACK has no default extensions.

The name for *packed_file* must be different from the *exe_file* filename. Although it is possible to fool EXEPACK into creating a packed file with the same name by specifying a different but equivalent pathname for the output file, the resulting packed file will probably be damaged. If the packed file is to replace the original .EXE file, a different name should be specified for the packed file; then the input file should be deleted and the packed file renamed with the name of the original file.

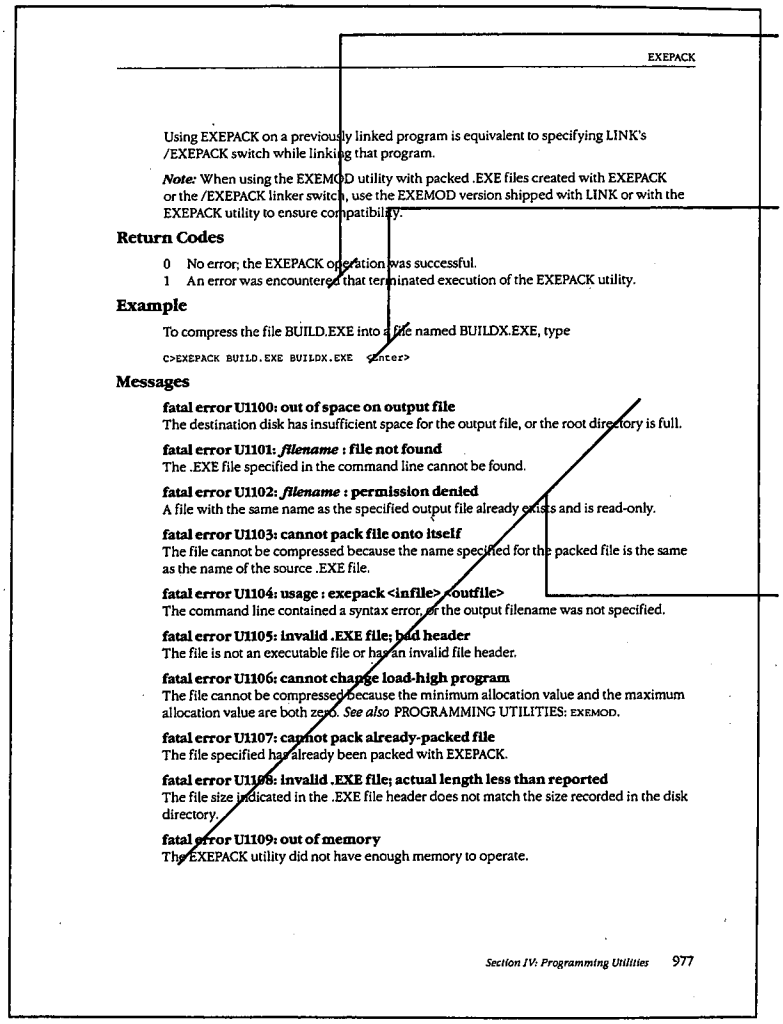
When EXEPACK is used to compress an executable overlay file or a program that calls overlays, the packed file should be renamed with its original name before use to avoid interruption by the overlay-manager prompt.

The effects of EXEPACK depend on program characteristics. Most programs can be processed with EXEPACK to occupy significantly less disk space. Programs thus compressed also load for execution more quickly. Occasionally programs (particularly small ones) actually become larger after processing with EXEPACK; in such cases the file produced by EXEPACK should be discarded. Microsoft Windows programs or programs to be debugged under DEBUG, SYMDEB, or CodeView should not be compressed with EXEPACK.

976 *The MS-DOS Encyclopedia*

BELOW WHERE
A brief explanation of each command parameter and switch. Filenames are always listed first, followed by the switches in alphabetic order. Any special position required for a filename or switch is shown in the syntax line and noted in the explanation.

DESCRIPTION
A detailed description of the utility, including a full explanation of default values, possible interactions of command parameters and options, useful background information, and any applicable warnings.



EXEPACK

Using EXEPACK on a previously linked program is equivalent to specifying LINK's /EXEPACK switch while linking that program.

*Note:* When using the EXEMOD utility with packed .EXE files created with EXEPACK or the /EXEPACK linker switch, use the EXEMOD version shipped with LINK or with the EXEPACK utility to ensure compatibility.

**Return Codes**

0 No error; the EXEPACK operation was successful.
1 An error was encountered that terminated execution of the EXEPACK utility.

**Example**

To compress the file BUILD.EXE into a file named BUILDX.EXE, type

```
C>EXEPACK BUILD.EXE BUILDX.EXE <Enter>
```

**Messages**

**fatal error U1100: out of space on output file**
The destination disk has insufficient space for the output file, or the root directory is full.

**fatal error U1101: *filename* : file not found**
The .EXE file specified in the command line cannot be found.

**fatal error U1102: *filename* : permission denied**
A file with the same name as the specified output file already exists and is read-only.

**fatal error U1103: cannot pack file onto itself**
The file cannot be compressed because the name specified for the packed file is the same as the name of the source .EXE file.

**fatal error U1104: usage : exepack <infile> <outfile>**
The command line contained a syntax error, or the output filename was not specified.

**fatal error U1105: invalid .EXE file; bad header**
The file is not an executable file or has an invalid file header.

**fatal error U1106: cannot change load-high program**
The file cannot be compressed because the minimum allocation value and the maximum allocation value are both zero. *See also* PROGRAMMING UTILITIES: EXEMOD.

**fatal error U1107: cannot pack already-packed file**
The file specified has already been packed with EXEPACK.

**fatal error U1108: invalid .EXE file; actual length less than reported**
The file size indicated in the .EXE file header does not match the size recorded in the disk directory.

**fatal error U1109: out of memory**
The EXEPACK utility did not have enough memory to operate.

*Section IV: Programming Utilities* 977

RETURN CODES
Exit codes returned by the utility (if any) that can be tested in a batch file or by another program.

EXAMPLES
One or more examples of the utility at work, including examples of the resulting output where appropriate. User entry appears in color; do not type the prompt, which appears in black. Press the Enter key (labeled Return on some keyboards) as directed at the end of each command line.

MESSAGES
An alphabetic list of messages that may be displayed when the utility is used. Following each message is a brief explanation of the condition that produces the message and, where appropriate, any action that should be taken.

# CREF

Generate Cross-Reference Listing

## Purpose

Produces a cross-reference listing of all symbols in an assembly-language program. The CREF utility is supplied with the Microsoft Macro Assembler (MASM). This documentation describes CREF version 4.0.

## Syntax

CREF

or

CREF *crf_file*[;]

or

CREF *crf_file,ref_file*

where:

*crf_file* is the input file previously produced by MASM (default extension = .CRF).
*ref_file* is the output ASCII text file to be created (default extension = .REF).

## Description

The CREF utility processes a file produced by MASM and generates an ASCII cross-reference listing in a file on disk or directly on a character device (such as a printer). The output file contains an alphabetic list of the symbols in the assembled program, including the line number of each reference to the symbol and the total number of symbols in the program. A pound sign (#) follows the line number of the reference that defines the symbol.

The *crf_file* has the default extension .CRF. It is produced by providing MASM with a file-name other than NUL in the cross-reference position in the command line, by responding to the *Cross-reference:* prompt, or by including the /C switch in the MASM command line or at any MASM prompt. An assembly source listing file (.LST) must also be requested in the MASM command line or in response to the MASM prompts in order to generate a valid .CRF file.

If a semicolon follows the *crf_file* parameter in the CREF command, the resulting *ref_file* containing the cross-reference listing is given the same drive and pathname as *crf_file*, with a .REF extension. If the optional *ref_file* parameter is present, it can consist of any pathname with an optional extension (default is .REF). The cross-reference listing can be sent directly to a character device, rather than to a file, by specifying a valid character device name (such as PRN) in the *ref_file* position.

If the CREF utility is run without any parameters or with some parameters missing, the CREF utility prompts the operator for the necessary information.

## Return Codes

0 No error; the CREF operation was successful.
1 An error was encountered that terminated execution of the CREF utility.

## Examples

To process the file MENUMGR.CRF (created during assembly of MENUMGR.ASM) into the cross-reference file MENUMGR.REF, type

```
C>CREF MENUMGR;  <Enter>
```

To process the file MENUMGR.CRF and assign the name MENU.REF to the resulting cross-reference file, type

```
C>CREF MENUMGR,MENU  <Enter>
```

To process the file MENUMGR.CRF and send the cross-reference listing directly to the printer, type

```
C>CREF MENUMGR,PRN  <Enter>
```

To run the CREF program in interactive mode, type

```
C>CREF  <Enter>
```

The following is an example of an interactive CREF session:

```
C>CREF  <Enter>
Microsoft (R) Cross Reference Utility  Version 4.00
Copyright (C) Microsoft Corp 1981, 1983, 1984, 1985.  All rights reserved.

Cross-reference [.CRF]: MENUMGR  <Enter>
Listing [MENUMGR.REF]:  <Enter>

9 Symbols

C>
```

The following sequence of commands produces the cross-reference listing HELLO.REF from the assembly-language source file HELLO.ASM:

```
C>MASM HELLO,HELLO,HELLO,HELLO  <Enter>
C>CREF HELLO;  <Enter>
```

Contents of the file HELLO.ASM:

```
        name    hello
        page    55,132
        title   HELLO.ASM - print Hello on terminal
;
; HELLO.COM utility to demonstrate CREF listing
;
cr      equ     0dh             ;ASCII carriage return
lf      equ     0ah             ;ASCII linefeed

cseg    segment para public "CODE"

        org     100h

        assume  cs:cseg,ds:cseg,es:cseg,ss:cseg

print   proc    near
        mov     dx,offset message
        mov     ah,9            ;print the string "Hello"
        int     21h
        mov     ax,4c00h        ;exit to MS-DOS
        int     21h             ;with "return code" of zero
print   endp

message db      cr,lf,'Hello!',cr,lf,'$'

cseg    ends

        end     print
```

Contents of the file HELLO.REF:

```
Microsoft Cross-Reference  Version 4.00              Mon Sep 07 23:31:21 1987
HELLO.ASM - print Hello on terminal

  Symbol Cross-Reference           (# is definition)                    Cref-1

CODE . . . . . . . . . . . . . .   10
CR . . . . . . . . . . . . . . .    7      7#    24      25
CSEG . . . . . . . . . . . . . .   10     10#    14      14      14      14      27

LF . . . . . . . . . . . . . . .    8      8#    24      25

MESSAGE. . . . . . . . . . . . .   17     24     24#

PRINT. . . . . . . . . . . . . .   16     16#    29


 6 Symbols
```

## Messages

**can't open cross-reference file for reading**
The pathname or drive specified for the input .CRF file is invalid or does not exist.

**can't open listing file for writing**
A write error has halted the creation of the .REF listing file. This indicates that the disk is full or write-protected, that the specified output file is read-only, or that the specified device is not available.

**cref has no switches**
A switch was specified in the command line; CREF has no optional switches.

**DOS 2.0 or later required**
CREF does not work with versions of MS-DOS earlier than 2.0.

**extra file name ignored**
More than two filenames were specified in the command line. The CREF utility generates the cross-reference listing using the first two filenames specified.

**line invalid, start again**
No .CRF file was specified in the command line or at the prompt. Specify a valid .CRF file at the prompt following this message.

**out of heap space**
Memory is insufficient to process the .CRF file. Remove memory-resident programs and shells or add more memory.

**premature eof**
The input file specified is damaged or is not a valid .CRF file.

**read error on stdin**
A Control-Z was received from the keyboard or a redirected file and has halted CREF.

# EXE2BIN

Convert .EXE File to Binary-Image File

## Purpose

Converts an executable file in the .EXE format to a memory-image file in binary format. The EXE2BIN utility is supplied with the MS-DOS distribution disks.

## Syntax

EXE2BIN *exe_file* [*bin_file*]

where:

| | |
|---|---|
| *exe_file* | is the .EXE-format file to be converted (default extension = .EXE). |
| *bin_file* | is the name to be given to the converted file (default extension = .BIN). |

## Description

The .EXE executable program files produced by the Microsoft Object Linker (LINK) contain a special header and a relocation table as well as the program code and data. The EXE2BIN utility can be used to convert a .EXE file to a .COM executable file, which is an absolute memory image of the program to be executed and does not contain a special header or relocation table. The EXE2BIN utility can also be used to convert .EXE files with an origin of zero (such as installable MS-DOS device drivers) to pure memory-image files. Files in memory-image format (a common format for device drivers and for programs to be placed in ROM for execution) usually have a .BIN or .SYS extension.

To convert a .EXE program to a binary-image file, the following are required:

- The program must be a valid .EXE file produced by LINK.
- The program can contain only one segment and cannot contain a declared stack segment.
- The program code and data portion of the .EXE file must be less than 64 KB.

To convert a .EXE program to an executable .COM file, the following are required:

- The origin of the program must be 0100H, which must also be specified as the entry point.
- The program code and data portion of the .EXE file must be less than 65227 bytes (64 KB minus 256 bytes used by the program segment prefix minus 2 bytes initially placed on the stack).
- The program must not include any FAR references.

***Note:*** Many compilers cannot create programs that can be converted to .COM files. Check the compiler documentation for specific information concerning executable .COM files.

The *exe_file* parameter in the command line can have any filename and can include a drive and path; the default extension is .EXE. The optional *bin_file* parameter can also contain any filename and a drive and path; the default extension is .BIN. If no path is specified with the *bin_file* parameter, the output file is given the same drive and path as the *exe_file*. If no *bin_file* parameter is supplied, the output file is given the same name as the *exe_file*, with the extension .BIN.

If the program in the .EXE file requires segment fixups (that is, if the program contains instructions requiring segment relocation, which would ordinarily be done by the MS-DOS loader using the .EXE file's relocation table), EXE2BIN prompts for a base segment address. When segment fixups are necessary, the resulting program is not relocatable and must be loaded at the given location to be executed; the MS-DOS loader cannot load the program.

## Examples

To convert the file HELLO.EXE to the file HELLO.BIN, type

```
C>EXE2BIN HELLO  <Enter>
```

To convert the file CLEAN.EXE, which has an origin of 0100H and meets the requirements for an executable .COM file, to the file CLEAN.COM, type

```
C>EXE2BIN CLEAN.EXE CLEAN.COM  <Enter>
```

To convert the file ASYNCH.EXE, produced by assembling and linking the device-driver source file ASYNCH.ASM, to the installable device-driver file ASYNCH.SYS, type

```
C>EXE2BIN ASYNCH.EXE ASYNCH.SYS  <Enter>
```

## Messages

**File cannot be converted**
The program to be converted has one of the following problems: The program has an origin of 0100H but a different entry point; the program requires segment fixups; the program code and data are larger than 64 KB; the program has more than one declared segment; or the file is not a valid .EXE-format file.

**File creation error**
EXE2BIN cannot create the output file because a read-only file with the same name already exists, because the specified directory is full, or because the specified disk is full, write-protected, or unreadable.

**File not found**
The file does not exist or the incorrect path was given.

**Fixups needed - base segment (hex):**
The .EXE-format file contains segment references that would ordinarily be relocated by the .EXE file loader. Specify the absolute segment address at which the converted module will be executed.

**Incorrect DOS version**
The version of EXE2BIN is not compatible with the version of MS-DOS that is running.

**Insufficient disk space**
The destination disk has insufficent space to create the memory-image output file.

**Insufficient memory**
Not enough memory is available to run EXE2BIN.

**WARNING - Read error in EXE file.**
**Amount read less than size in header.**
The file size given in the .EXE header is inconsistent with the actual size of the file.

# EXEMOD

Modify .EXE File Header

## Purpose

Allows inspection or modification of the fields in a .EXE file header. The EXEMOD utility is supplied with the Microsoft Macro Assembler (MASM), C Compiler, FORTRAN Compiler, and Pascal Compiler. This documentation describes EXEMOD version 4.02.

## Syntax

EXEMOD *exe_file*[/H]

or

EXEMOD *exe_file*[/STACK *n*][/MAX *n*][/MIN *n*]

where:

| | |
|---|---|
| *exe_file* | is the name of an executable program in .EXE format (the extension .EXE is assumed). |
| /H | displays the values in the file's header. |
| /STACK *n* | modifies the size of the program's stack segment to *n* (hexadecimal) bytes. |
| /MAX *n* | sets the maximum memory allocation for the program to *n* (hexadecimal) paragraphs. |
| /MIN *n* | sets the minimum memory allocation for the program to *n* (hexadecimal) paragraphs. |

***Note:*** Switches can be either uppercase or lowercase and can be preceded by a dash (-) instead of a forward slash (/).

## Description

Programs that are executable under MS-DOS can be in one of two file formats: .COM, which is an absolute image of the file to be executed and limits the program size to 65227 bytes (64 KB minus 256 bytes used by the program segment prefix minus 2 bytes initially placed on the stack); or .EXE, which allows a program of any size to be loaded and has a special header containing information about the program's entry point, stack size, and memory requirements, plus a relocation table.

The EXEMOD utility can be used to display or modify those fields of a .EXE program header that control the size of the stack segment and the amount of memory allocated to the program when MS-DOS loads the program into the transient program area for execution.

The /STACK*n* switch controls the number of bytes in the program's STACK segment by setting the initial SP to the hexadecimal value specified. The minimum paragraph allocation value is adjusted if necessary. The EXEMOD /STACK*n* switch should be used only with programs compiled by Microsoft C version 3.0 or later, Microsoft Pascal version 3.3

or later, or Microsoft FORTRAN version 3.0 or later. Use of the /STACK*n* switch with a program developed with another compiler can cause the program to fail or cause EXEMOD to return an error message.

The /MAX*n* switch specifies the maximum number of additional paragraphs of memory to allocate for use by the program. The /MIN*n* switch specifies the minimum number of paragraphs of memory, in addition to the size of the program itself and its stack and data segments, that are required for the program to execute. If enough memory exists to satisfy the minimum additional paragraphs requested but not enough exists to satisfy the maximum, MS-DOS allocates all available memory to the program.

To display the current memory allocation and stack size values from a .EXE file's header, the /H switch can be used or the file's name can be entered as the only parameter in the command line.

When EXEMOD is used on a previously packed .EXE file (a file that was processed by EXEPACK or linked with the /EXEPACK switch), the values set or displayed in the file's header are the values that will apply after the file is expanded at load time. EXEMOD displays a message advising the user that the file being modified was previously packed.

The EXEMOD switches /MAX*n* and /STACK*n* correspond to the Microsoft Object Linker's /CPARMAXALLOC:*n* and /STACK:*n* switches, respectively. *See* PROGRAMMING UTILITIES: LINK.

## Return Codes

0 No error; EXEMOD operation was successful.
1 An error was encountered that terminated execution of the EXEMOD program.

## Examples

To display the values in the file header of the DUMP.EXE program, type

```
C>EXEMOD DUMP.EXE  <Enter>
```

or

```
C>EXEMOD DUMP.EXE /H  <Enter>
```

The EXEMOD utility displays the following:

```
Microsoft (R) EXE File Header Utility  Version 4.02
Copyright (C) Microsoft Corp 1985.  All rights reserved.
DUMP.EXE                          (hex)           (dec)

.EXE size (bytes)                   580            1408
Minimum load size (bytes)           383             899
Overlay number                        0               0
Initial CS:IP                 0000:0000
Initial SS:SP                 0034:0040              64
Minimum allocation (para)             5               5
Maximum allocation (para)          FFFF           65535
Header size (para)                   20              32
Relocation table offset              20              32
Relocation entries                    1               1
```

To change the size of the STACK segment for the DUMP.EXE program to 400H (1024) bytes, type

```
C>EXEMOD DUMP.EXE /STACK 400  <Enter>
```

EXEMOD displays the message

```
EXEMOD : warning U4051: minimum allocation less than stack; correcting minimum
```

## Messages

**error U1050: usage : exemod file [-/h] [-/stack n] [-/max n] [-/min n]**
An error was detected in the EXEMOD command line.

**error U1051: invalid .EXE file : bad header**
The file is not an executable file or has an invalid file header.

**error U1052: invalid .EXE file : actual length less than reported**
The file size indicated in the .EXE file header does not match the size recorded in the disk directory.

**error U1053: cannot change load-high program**
The header of the file cannot be modified because the minimum allocation value and the maximum allocation value are both zero.

**error U1054: file not .EXE**
The file specified does not have a .EXE extension.

**error U1055: *filename* : cannot find file**
The .EXE file specified in the command line cannot be found.

**error U1056: *filename* : permission denied**
The .EXE file specified in the command line is read-only.

**warning U4050: packed file**
The specified file is a packed file; that is, it was previously processed with the EXEPACK utility or was linked with the /EXEPACK switch. This is an informational message only; EXEMOD still modifies the file. The header values displayed are the values that will apply after the packed value is expanded at load time.

**warning U4051: minimum allocation less than stack; correcting minimum**
The minimum allocation value is not large enough to accommodate the stack; the minimum allocation value is adjusted. This is an informational message only.

**warning U4052: minimum allocation greater than maximum; correcting maximum**
If the minimum allocation value is greater than the maximum allocation value, the maximum value is adjusted. This is an informational message only.

# EXEPACK

Compress .EXE File

## Purpose

Compresses an executable .EXE program file so that it requires less space on the disk. The EXEPACK utility is supplied with the Microsoft Macro Assembler (MASM), C Compiler, FORTRAN Compiler, and Pascal Compiler. This documentation describes EXEPACK version 4.04.

## Syntax

EXEPACK *exe_file packed_file*

where:

| | |
|---|---|
| *exe_file* | is the name of the executable .EXE program file to be compressed. |
| *packed_file* | is the name of the compressed program file. |

## Description

The EXEPACK utility compresses an executable .EXE program by packing sequences of identical bytes and optimizing the relocation table. The EXEPACK utility is not compatible with versions of MS-DOS earlier than 2.0.

The *exe_file* parameter specifies the name of the program file produced by the Microsoft Object Linker (LINK) and must contain the extension .EXE. The *packed_file* parameter specifies the name and extension of the resulting compressed file. EXEPACK has no default extensions.

The name for *packed_file* must be different from the *exe_file* filename. Although it is possible to fool EXEPACK into creating a packed file with the same name by specifying a different but equivalent pathname for the output file, the resulting packed file will probably be damaged. If the packed file is to replace the original .EXE file, a different name should be specified for the packed file; then the input file should be deleted and the packed file renamed with the name of the original file.

When EXEPACK is used to compress an executable overlay file or a program that calls overlays, the packed file should be renamed with its original name before use to avoid interruption by the overlay-manager prompt.

The effects of EXEPACK depend on program characteristics. Most programs can be processed with EXEPACK to occupy significantly less disk space. Programs thus compressed also load for execution more quickly. Occasionally programs (particularly small ones) actually become larger after processing with EXEPACK; in such cases the file produced by EXEPACK should be discarded. Microsoft Windows programs or programs to be debugged under DEBUG, SYMDEB, or CodeView should not be compressed with EXEPACK.

Using EXEPACK on a previously linked program is equivalent to specifying LINK's /EXEPACK switch while linking that program.

***Note:*** When using the EXEMOD utility with packed .EXE files created with EXEPACK or the /EXEPACK linker switch, use the EXEMOD version shipped with LINK or with the EXEPACK utility to ensure compatibility.

## Return Codes

0 No error; the EXEPACK operation was successful.
1 An error was encountered that terminated execution of the EXEPACK utility.

## Example

To compress the file BUILD.EXE into a file named BUILDX.EXE, type

```
C>EXEPACK BUILD.EXE BUILDX.EXE  <Enter>
```

## Messages

**fatal error U1100: out of space on output file**
The destination disk has insufficient space for the output file, or the root directory is full.

**fatal error U1101: *filename* : file not found**
The .EXE file specified in the command line cannot be found.

**fatal error U1102: *filename* : permission denied**
A file with the same name as the specified output file already exists and is read-only.

**fatal error U1103: cannot pack file onto itself**
The file cannot be compressed because the name specified for the packed file is the same as the name of the source .EXE file.

**fatal error U1104: usage : exepack <infile> <outfile>**
The command line contained a syntax error, or the output filename was not specified.

**fatal error U1105: invalid .EXE file; bad header**
The file is not an executable file or has an invalid file header.

**fatal error U1106: cannot change load-high program**
The file cannot be compressed because the minimum allocation value and the maximum allocation value are both zero. *See also* PROGRAMMING UTILITIES: EXEMOD.

**fatal error U1107: cannot pack already-packed file**
The file specified has already been packed with EXEPACK.

**fatal error U1108: invalid .EXE file; actual length less than reported**
The file size indicated in the .EXE file header does not match the size recorded in the disk directory.

**fatal error U1109: out of memory**
The EXEPACK utility did not have enough memory to operate.

**fatal error U1110: error reading relocation table**
The file cannot be compressed because the relocation table cannot be found or is invalid.

**fatal error U1111: file not suitable for packing**
The file could not be packed because the packed load image of the specified file was larger than the unpacked load image.

**fatal error U1112: *filename* : unknown error**
An unknown system error occurred while the specified file was being processed.

**warning U4100: omitting debug data from output file**
EXEPACK has stripped all symbolic debug information from the output file.

# LIB

Library Manager

## Purpose

Creates or modifies an object module library file. The LIB utility is supplied with the Microsoft Macro Assembler (MASM), C Compiler, FORTRAN Compiler, and Pascal Compiler. This documentation describes LIB version 3.06.

## Syntax

LIB

or

LIB *library_file* [/PAGESIZE:*n*] [*operation*][,[*list_file*][,[*new_library_file*]]] [;]

or

LIB @*response_file*

where:

*library_file* is the name of the object module library file to be created or modified (default extension = .LIB).

/PAGESIZE:*n* is the page size of the library file and must immediately follow *library_file* if used; *n* is a power of 2 between 16 and 32768, inclusive (default = 16). Can be abbreviated /P:*n*.

*operation* is one or more library manipulations to be performed. Each *operation* is specified as a code followed by an object module name (case is not significant):

| | |
|---|---|
| +*name* | Add object module or another library to library. |
| −*name* | Delete object module from library. |
| −+*name* | Replace object module in library. |
| **name* | Copy object module from library to object file. |
| −**name* | Copy object module to object file and then delete object module from library. |

*list_file* is the name of the file or character device to receive the cross-reference listing for the library file (default = NUL device).

*new_library_file* is the name to be assigned to the modified object module library file. (The default name is the same as *library_file*; if the default is used, the original *library_file* is renamed with the extension .BAK.)

*response_file* is the name of a text file containing LIB parameters in the same order in which they are supplied if entered interactively. The name of the response file must be preceded by the @ symbol.

## Description

The Microsoft Library Manager (LIB) creates and modifies library files, checks existing library files for consistency, and prints listings of the contents of library files. The LIB utility does not work with versions of MS-DOS earlier than 2.0.

A library file consists of relocatable object modules that are indexed by their names and public symbols. The Microsoft Object Linker (LINK) uses these files during the creation of an executable (.EXE) program to resolve external references to routines and variables contained in other object modules.

The *library_file* parameter specifies the name of the object module library file to be created or modified. This parameter is required; if it is not included, LIB prompts for it. The default extension for a library file is .LIB.

The /PAGESIZE:*n* switch (abbreviated /P:*n*) sets the page size (in bytes) for a new library file or changes the page size of an existing library file. The value of *n* must be a power of 2 between 16 and 32768, inclusive. The default is 16 for a new library file; for an existing library file, the default is the current page size. Because the index to a library file is contained in a fixed number of pages, setting a larger page size increases the number of index entries (and thus the number of object modules) that a library file can contain but results in more wasted disk space (an average of half a library page per object module).

The *operation* parameter specifies one or more relocatable object modules to add to, replace in, copy from, move from, or delete from *library_file*. Each operation is represented by a code specifying the type of operation, followed by the object module name. When an object module is copied or moved from the library file, the drive and pathname of the object module are set to the default drive, current directory, and specified module name, and the extension of the object module defaults to .OBJ. When an object module is added or replaced, LIB assumes a default extension of .OBJ.

The operation +*name* adds the object module in the file *name*.OBJ to the library file. This operation can also be used to add the contents of another entire object module library file to the library file being updated, in which case the extension .LIB must be included in *name*. The operation –*name* deletes the object module *name* from the library. The operation –+*name* deletes the object module *name* from the library file and replaces it with the contents of the file *name*.OBJ. The operation **name* copies the object module *name* from the library file into the file *name*.OBJ, which LIB creates in the current directory. The operation –**name* also copies the object module *name* from the library file into a .OBJ file but then deletes the module from the library file. (Although *name* must have exactly the same spelling as the name in the library's reference listing, case is not significant.)

***Note:*** LIB does not actually delete object modules from the specified library file. Instead, it marks the selected object modules for deletion, creates a new library file, and copies only

the modules not marked for deletion into the new file. Thus, if LIB is terminated for any reason, the original file is not lost. Enough space must be available on the disk for both the original library file and the copy.

The *list_file* parameter specifies the file or character device to receive a reference listing for the library file. Any valid drive, pathname, and extension or any valid character device, such as PRN, is permitted (default = NUL). If this parameter is omitted, no listing is generated.

The reference listing consists of two tables. The first table contains all the public symbols in the object modules in the library, listed alphabetically, with each symbol followed by the name of the object module in which it is referenced. The second table contains the names of all the object modules, listed alphabetically, with each name followed by the offset from the start of the library file, the code and data size, and an alphabetic listing of the public symbols in that object module.

The *new_library_file* parameter specifies the name for the modified library file that is created. If this parameter is omitted, LIB gives the modified library file the same name as the original library file, and the original library file is renamed with a .BAK extension. When a new library file is being created, this parameter is not necessary.

When the command line is used to supply LIB with filenames and switches, typing a semicolon character (;) after any parameter (except *library_file*) causes LIB to use the default values for the remaining parameters. If a semicolon is entered after *library_file*, LIB simply checks the file for consistency and usability. (This is seldom necessary, because LIB checks each object module for consistency before adding it to the library.)

If the LIB command is entered without any parameters, LIB prompts the user for each parameter needed. If there are too many operations to fit on one line, the line can be ended with the ampersand character (&), causing LIB to repeat the *Operations:* prompt. If any response except *library_file* is terminated with a semicolon character, LIB uses the default values for the remaining filenames. When the *library_file* parameter is followed by a semicolon or a semicolon is entered at the *Operations:* prompt, LIB takes no action except to verify that the contents of the specified file are consistent and usable.

The *response_file* parameter allows the automation of complex LIB sessions involving many files. A response file contains ASCII text that corresponds line for line to the responses that are entered in a normal interactive LIB session, in the form

*library_file*[/P:*n*]
[Y]
[*operations*]
[*list_file*]
[*new_library_file*][;]

The response file name must be preceded in the command line by the at symbol (@) and can also be preceded by a path and/or drive letter. If *library_file* is a new file, the letter *Y*

must appear by itself on the second line of the response file to approve the creation of a library file. The last line of the response file must end with a semicolon or a carriage return. (LIB ignores any lines following a semicolon.) If all the parameters required by LIB are not present in the response file or the response file does not end with a semicolon, LIB prompts the user for the missing information.

## Return Codes

0 No error; LIB operation was successful.
1 An error that terminated execution of the LIB utility was encountered.

## Examples

To create a library file named MYLIB.LIB and insert the object files VIDEO.OBJ, COMM.OBJ, and DOSINT.OBJ, type

```
C>LIB MYLIB +VIDEO +COMM +DOSINT;  <Enter>
```

To print a listing of the object modules in the library file MYLIB.LIB, type

```
C>LIB MYLIB,PRN  <Enter>
```

If the LIB command is entered without parameters, the user is prompted for the necessary information. For example, if the user wanted to add the module VIDEO.OBJ to the library file SLIBC.LIB, produce a reference listing in the file SLIBC.LST, and produce a new output library file named SLIBC2.LIB, the following dialogue would take place:

```
C>LIB  <Enter>

Microsoft (R) Library Manager  Version 3.06
Copyright (C) Microsoft Corp 1983, 1984, 1985, 1986.  All rights reserved.

Library name: SLIBC  <Enter>
Operations: +VIDEO  <Enter>
List file: SLIBC.LST  <Enter>
Output library: SLIBC2  <Enter>
```

## Messages

***filename*: cannot access file.**
LIB is unable to access an object module specified in a response file, in the command line, or at the *Operations:* prompt.

***filename*: cannot create extract file**
The object module cannot be copied or moved from the library file into a separate disk file called *filename* because the root directory or disk is full or because *filename* already exists and is read-only.

***filename*: cannot create listing**
The list file specified in the response file, in the command line, or at the *List file:* prompt cannot be created because the root directory or disk is full or because *filename* already exists and is read-only.

***filename*: invalid format (*xxxx*); file ignored.**
The hexadecimal signature byte or word *xxxx* of the specified file was not one of the following recognized types: Microsoft library, Intel library, Microsoft object, or XENIX archive.

***filename*: invalid library header.**
The input library file either is not a library file or is damaged.

***filename*: invalid library header; file ignored.**
The input library file is in the wrong format.

***modulename*: invalid object module near *location***
The specified object module has an invalid format near the hexadecimal offset indicated.

***modulename*: module not in library; ignored**
The object module specified in the response file, in the command line, or at the *Operations:* prompt is not in the specified input library file.

***modulename*: module redefinition ignored**
An object module was specified to be added to a library file but an object module with the same name was already in the library file, or the same object module was specified twice in an add operation in the command line.

***number*: page size too small; ignored**
The size specified with a /P:*n* switch must be a power of 2 between 16 and 32768 bytes, inclusive.

***symbol* (*modulename*) : symbol redefinition ignored**
The specified symbol was defined in more than one module. Only the first definition of a symbol is accepted. All redefinitions are ignored.

**cannot create new library**
The root directory is full, or a library file with the same name already exists and is read-only.

**cannot open response file**
The specified response file cannot be found or does not exist.

**cannot rename old library**
The old library file cannot be renamed with a .BAK extension because such a file already exists and is read-only.

**cannot reopen library**
The old library file could not be reopened after it was renamed with the .BAK extension. This error usually indicates damage to the operating system or to the disk directory structure.

**comma or new line missing**
A comma or carriage return was expected in the command line but was not found.

**Do not change diskette in drive *X*:**
LIB may have placed important temporary files on the specified disk. Do not remove the disk until the LIB operation is complete or these files may be lost.

**error writing to cross-reference file**
The disk or root directory is full.

**error writing to new library**
The new library file cannot be created because the disk is full.

**free: not allocated**
This is a serious problem. Note the circumstances of the failure and notify Microsoft Corporation.

**insufficient memory**
Not enough memory is available in the transient program area for LIB to successfully perform the requested operations.

**internal failure**
This is a serious problem. Note the circumstances of the failure and notify Microsoft Corporation.

**Library does not exist. Create?**
The specified *library_file* does not exist on disk. Respond with *Y* to create the library file; respond with *N* to terminate the LIB utility.

**mark: not allocated**
This is a serious problem. Note the circumstances of the failure and notify Microsoft Corporation.

**option unknown**
The command line included a switch other than /P:*n*.

**output-library specification ignored**
An output library file was specified in addition to a new library file. This is only a warning. The output library file specification will be disregarded.

**page size too small**
The page size of an input library file was less than 16 bytes, indicating a damaged or otherwise invalid .LIB file. *See* LIB message *number*: page size too small; ignored.

**syntax error**
The command line included an invalid parameter or switch.

**syntax error: illegal file specification**
A command operator (such as *, –, or +) was given without an object module name.

**syntax error: illegal input**
The command line included an invalid parameter or switch.

**syntax error: option name missing**
The command line included a forward slash (/) that was not followed by P:*n*.

**syntax error: option value missing**
The /P switch was not followed by the page size value in bytes.

**terminator missing**
Either a control character (such as Control-Z) was specified at the *Output library:* prompt or the response file line that corresponds to LIB's *Output library:* prompt was not terminated by a carriage return or semicolon.

**too many symbols**
The maximum number of public symbols allowed in a library file has been exceeded. The limit for all object modules (combined) is 4609.

**unexpected end-of-file on command input**
The response file did not include all the necessary LIB parameters.

**write to extract file failed**
The destination disk has insufficient space for the complete object module, or the root directory is full.

**write to library file failed**
The destination disk has insufficient space to create the new library file, or the root directory is full.

# LINK

Create .EXE File

## Purpose

Combines relocatable object modules into an executable (.EXE) file. The Microsoft Object Linker (LINK) is supplied with the Microsoft Macro Assembler (MASM), C Compiler, Pascal Compiler, and FORTRAN Compiler. This documentation describes LINK version 3.50.

## Syntax

LINK

or

LINK *obj_file*[+*obj_file*...][,[*exe_file*]][,[*map_file*]][,[*library*[+*library*...]]] [*options*] [;]

or

LINK @*response_file*

where:

*obj_file* is the name of a file containing a relocatable object module produced by MASM or by a high-level-language compiler (default extension = .OBJ).

*exe_file* is the name of the executable file to be produced by LINK (default extension = .EXE).

*map_file* is the name of the file or character device to receive a listing of the names, load addresses, and lengths of the segments in *exe_file* (default = NUL device; default extension = .MAP).

*library* is the name of an object module library to be searched to resolve external references in the object file(s) (default extension = .LIB).

*response_file* is the name of a text file containing LINK parameters in the order in which they are supplied during an interactive LINK session.

*options* specifies one or more of the following switches. Switches can be either uppercase or lowercase.

- /CP:*n* (/CPARMAXALLOC:*n*) Sets the maximum number of extra memory paragraphs required by *exe_file* (default = 65535).
- /DS (/DSALLOCATE) Loads the data in DGROUP at the high end of the data segment.
- /DO (/DOSSEG) Arranges segments according to the Microsoft language segment-ordering convention.
- /E (/EXEPACK) Compresses repetitive sequences of bytes and optimizes *exe_file*'s relocation table.

*(more)*

| | |
|---|---|
| /HI | (/HIGH) Causes *exe_file* to be loaded as high as possible in memory when *exe_file* is executed. |
| /HE | (/HELP) Lists LINK options on the screen. No other switches or filenames should be used with this switch. |
| /LI | (/LINENUMBERS) Copies line-number information (if available) from *obj_file* to *map_file*. If a map file was not specified, this switch creates one. |
| /M | (/MAP) Copies a list of all public symbols declared in *obj_file* to *map_file*. If a map file was not specified, this switch creates one. |
| /NOD | (/NODEFAULTLIBRARYSEARCH) Causes LINK to ignore any library names inserted in the object file by the language compiler. |
| /NOG | (/NOGROUPASSOCIATION) Causes LINK to ignore GROUP associations when assigning addresses. |
| /NOI | (/NOIGNORECASE) Causes LINK to be case sensitive when resolving external names. |
| /O:*n* | (/OVERLAYINTERRUPT:*n*) Overrides the interrupt number used by the overlay manager (0–255, default = 63, or 3FH). This switch should be used only when linking with a run-time module from a language compiler that supports overlays. |
| /P | (/PAUSE) Causes LINK to pause and prompt the user to change disks before writing the *exe_file*. |
| /SE:*n* | (/SEGMENTS:*n*) Sets the maximum number of segments that can be processed (1–1024, default = 128). |
| /ST:*n* | (/STACK:*n*) Sets the size of the *exe_file*'s stack segment to *n* bytes (1–65535). |

## Description

LINK combines relocatable object modules into an executable file in the .EXE format. LINK can be used with object files produced by any high-level-language compiler or assembler that supports the Microsoft object module format. *See* PROGRAMMING IN THE MS-DOS ENVIRONMENT: PROGRAMMING TOOLS: Object Modules; The Microsoft Object Linker.

The *obj_file* parameter, which is required, specifies one or more files containing relocatable object modules. If multiple object files are linked, their names should be separated by a plus operator (+) or a space. If an extension is not specified for an object file, LINK supplies the extension .OBJ. Some high-level-language compilers support partitioning of the executable program into a root segment and one or more overlay segments and include a special overlay manager in their libraries; when these compilers are used, the object modules that compose each overlay segment should be surrounded with parentheses in the LINK command line.

The *exe_file* parameter specifies the name of the executable file that is created by LINK. The default is the same filename as the first object file, but with the extension .EXE.

The *map_file* parameter designates the file or character device to receive LINK's listing of the name, load address, and length of each of *exe_file*'s segments. The map file also includes the names and load addresses of any groups in the program, the program entry point, and, if the /M switch is used, all public symbols and their addresses. If the /LI switch is used and if line numbers were inserted into *obj_file* by the compiler, the starting address of each *obj_file* program line is also copied to *map_file*. The default extension for a map file is .MAP. If the /M or /LI switch is used, a map file is created using the name of the specified .EXE file even if *map_file* is not specified. If neither the /M nor the /LI switch is used and *map_file* is not specified, no listing is created.

The *library* parameter specifies the object module library or libraries that will be searched to resolve external references after all the object files are processed. The default extension for library files is .LIB. Multiple library names should be separated by plus operators (+) or spaces. A maximum of 16 search paths can be specified in the LINK command line. If a library name is preceded by a drive and/or path, LINK searches only the specified location. If no drive or path precedes a library name, LINK searches for library files in the following order:

1. Current drive and directory
2. Any other library search paths specified in the command line, in the order they were entered
3. Directories specified in the LIB= environment variable, if one exists

In the following example, LINK searches only the \ALTLIB directory on drive A to find the library MATH.LIB. To find the library COMMON.LIB, LINK searches the current directory on the current drive, then the current directory on drive B, then directory \LIB on drive D, and finally, any directories named in the LIB environment variable.

```
C>LINK TEST,,TEST,A:\ALTLIB\MATH.LIB+COMMON+B:+D:\LIB\  <Enter>
```

If default libraries are specified within the object files through special records inserted by certain high-level-language compilers, those libraries will be searched *after* the libraries named in the command line or response file.

If the LINK command is entered without parameters, LINK prompts the user for each filename needed. The default response for each prompt (except the *obj_file* prompt) is displayed in square brackets and can be selected by pressing the Enter key. If there are too many *obj_file* or *library* names to fit on one line, the line can be terminated by entering a plus operator (+) and pressing the Enter key; LINK then repeats the prompt. If the user ends any response with a semicolon character (;), LINK uses the default values for the remaining fields.

When the command line contains filenames and switches, commas must be used to separate the *obj_file*, *exe_file*, *map_file*, and *library* parameters. If a filename is not supplied, a comma must be used to mark its place. If the user places a semicolon after any parameter in the command line, LINK terminates the command line at the semicolon and uses the default values for any remaining parameters.

The user can automate complex LINK sessions involving multiple files by creating a response file. The *response_file* parameter must be the name of an ASCII file that corresponds line for line to the responses that are entered in a normal interactive LINK session. The last line of the response file must end with a semicolon character (;) or a carriage return. If all parameters required by LINK are not present in the response file and the response file does not end with a semicolon or carriage return, LINK prompts the user for the missing information.

LINK supports many options that can be invoked by including a switch in the command line, as part of the response to a LINK prompt, or in a response file. To simplify this description, these switches are grouped according to their functions.

The /E, /HE, /NOD, /NOI, /P, and /SE:*n* switches affect LINK's general operation. The /E switch compresses repetitive sequences of bytes in *exe_file* and optimizes certain parts of the relocation table in *exe_file*'s header. The /E switch functions exactly like the EXEPACK utility.

***Note:*** The /E switch does not always save a significant amount of disk space and may even increase file size when used with small programs that have few load-time relocations or repeated characters. The Microsoft Symbolic Debugger (SYMDEB) utility cannot be used with packed files.

The /HE switch displays the available options on the screen. No other switches or filenames should be specified if the /HE switch is used. The /NOD switch causes LINK to ignore any default libraries that have been added to the object modules by the high-level-language compiler that produced the modules, thus restricting searches to those libraries specified in the command line or response file. The /NOI switch causes LINK to be case sensitive when resolving external references to symbols between object modules. The /NOI switch is typically used with object files created by high-level-language compilers that differentiate between uppercase and lowercase letters.

The /P switch causes LINK to pause and prompt the user before writing *exe_file* to disk, thus allowing the user to exchange the disk used during the linking operation for another that has more space available. The /SE:*n* switch controls the number of program segments processed by LINK. The *n* must be a decimal, octal, or hexadecimal number from 1 through 1024, inclusive (default = 128). Octal numbers must have a leading zero; hexadecimal numbers must begin with 0x.

The /M and /LI switches affect the production and contents of the optional map file. The /M switch creates a map file with the same name as *exe_file* or, if *exe_file* is not specified, with the same name as the first object file and the extension .MAP. The resulting map file includes a list of all public symbols and their addresses. The /LI switch also creates a map file and includes line-number information if available in the object file. (MASM and some high-level-language compilers do not insert line-number information into object files.)

The /D, /DO, /NOG, and /O:*n* switches affect the structure of the code in *exe_file*. Use of the /D switch places the data in DGROUP at the top (highest address) of the memory segment pointed to by the DS register, rather than at the bottom (the default). The /DO switch arranges the program segments according to a convention expected by all Microsoft language compilers: All segments with the class name CODE are placed first in the executable file; any other segments that do not belong to DGROUP are placed immediately after the CODE segments; all segments belonging to DGROUP are placed at the end of the file. The /NOG switch causes LINK to ignore group associations specified in the object modules when assigning addresses to data and code items; that is, segments that would ordinarily have been collected into the same physical memory segment because of their association within a GROUP are decoupled. The /NOG switch provides compatibility with LINK versions 2.02 and earlier and with early versions of Microsoft language compilers. The /O:*n* switch controls the interrupt number used by the resident overlay manager if the linked program includes overlays. The number *n* can be any decimal, octal, or hexadecimal number in the range 0 through 255 (default = 63, or 3FH). Octal numbers must have a leading zero; hexadecimal numbers must begin with 0x.

***Note:*** MASM and many high-level-language compilers do not include overlay managers in their libraries. Users should check their compiler documentation to determine if the /O:*n* switch can be used.

***Warning:*** Interrupt numbers that conflict with the software interrupts used to obtain MS-DOS or ROM BIOS services or with hardware interrupts assigned to peripheral device controllers should not be used in the /O:*n* switch.

The /C:*n*, /H, and /ST:*n* switches control the information in *exe_file*'s header that affects the behavior of the MS-DOS system loader when the file is read from the disk into RAM for execution. The /C:*n* switch sets the maximum number of 16-byte paragraphs of memory to be made available to the program when it is loaded into memory, in addition to the memory required to hold the program's code, data, and stacks; the default is 65535, which causes the program to be allocated all available memory. The /H switch causes the program to be loaded as high as possible in the transient program area (free memory), rather than as low as possible (the default). The /ST:*n* switch sets the stack size (in bytes) to be allocated for the program when it is loaded and overrides any stack segment size declarations in the original source code. The number *n* can be any decimal, octal, or hexadecimal number from 1 through 65535; however *n* must be large enough to accommodate any initialized data in the stack segment. Octal numbers must have a leading zero; hexadecimal numbers must begin with 0x. If the /ST:*n* switch is not used, LINK calculates a program's stack size, basing the size on the size of any stack segments given in the object files. The /C:*n* and /ST:*n* values in the *exe_file* header can be altered after linking by using the EXEMOD utility.

If LINK is unable to hold in RAM all the data it is processing, it creates a temporary disk file named VM.TMP (Virtual Memory) in the current directory of the default disk drive. If a floppy disk is in the default drive, LINK issues a warning message to prevent the user from changing disks until the LINK session is completed. After LINK finishes processing, it deletes the temporary file.

***Warning:*** Any file named VM.TMP that is already on the disk will be destroyed if LINK creates the temporary disk file.

## Return Codes

0 No errors or unresolved references were encountered during creation of *exe_file*.
1 A miscellaneous LINK error occurred that was not covered by the other return codes.
16 A data record was too large to process.
32 No object files were specified in the command line or response file.
33 The map file could not be created.
66 A COMMON area was declared that is larger than 65535 (one segment).
96 Too many libraries were specified.
144 An invalid object module (*obj_file*) was detected.
145 Too many TYPDEFs were found in the specified object modules.
146 Too many group, segment, or class names were found in one object module.
147 Too many segments were found in all the object modules combined, or too many segments were found in one object module.
148 Too many overlays were specified.
149 The size of a segment exceeded 65535.
150 Too many groups or GRPDEFs were found in one object module.
151 Too many external symbols were found in one object module.
177 The size of a group exceeded 65535.

## Examples

The simplest use of LINK is to process a single object file to produce an executable file, using all the default values. For example, to process the file SHELL.OBJ, create an executable file named SHELL.EXE, and search only the default libraries, type

```
C>LINK SHELL;  <Enter>
```

The semicolon after the filename causes LINK to use the default values for all other parameters.

To link three object files named SHELL.OBJ, VIDEO.OBJ, and DOSINT.OBJ into an executable file named SHELL.EXE and search the library DEVLIB.LIB on drive B before searching any default libraries, type

```
C>LINK SHELL+VIDEO+DOSINT,,,B:DEVLIB  <Enter>
```

If the LINK command is entered without parameters, LINK prompts the user for the necessary information. For example, the following interactive session links the file

MENUMGR.OBJ into the executable file MENUMGR.EXE, creates a map file named MENUMGR.MAP, and searches the math floating-point emulator library EM.LIB before any default libraries:

```
C>LINK  <Enter>

Microsoft (R) 8086 Object Linker  Version 3.05
Copyright (C) Microsoft Corp 1983,1984,1985.  All rights reserved.

Object Modules [.OBJ]: MENUMGR  <Enter>
Run File [MENUMGR.EXE]:  <Enter>
List File [NUL.MAP]:  MENUMGR  <Enter>
Libraries [.LIB]:  EM  <Enter>
```

## Messages

***filename* is not a valid library**
The file specified as an object module library either is corrupt or is not a library in the format created by the Microsoft LIB utility.

**About to generate .EXE file**
**Change diskette in drive *X* and press <ENTER>**
The /P switch was used in the command line. LINK is prompting the user to change disks before LINK creates the file containing the executable program.

**Ambiguous switch error: "*option*"**
A valid switch was not entered after a forward slash (/) in the command line.

**Array element size mismatch**
A FAR communal array was declared with two or more different array-element sizes (for example, once as an array of characters and once as an array of real numbers). This error occurs only with programs produced by the Microsoft C Compiler or other compilers that support FAR communal arrays; it does not occur with object files produced by MASM.

**Attempt to access data outside segment bounds**
A data record in an object module specified data extending beyond the end of a segment. This is a translator error. Note which compiler or assembler produced the invalid object module and notify Microsoft Corporation.

**Attempt to put segment *name* in more than one group in file *filename***
A segment was declared to be a member of two groups. Correct the source code and re-create the object modules.

**Bad value for cparMaxAlloc**
The value specified using the /C:*n* option is not in the range 1 through 65535.

**Cannot create temporary file**
The destination disk has insufficient space for the temporary file, or the root directory is full.

**Cannot find file *filename***
**Change diskette and press <ENTER>**
The specified object file cannot be found in the current drive.

**Cannot find library: *filename***
**Enter new file spec:**
The specified library file cannot be found or does not exist. Enter the correct drive letter, check the spelling of the filename and path, or make sure that the LIB environment variable has been set up properly.

**Cannot nest response files**
A response file was named within a response file. Revise the response file to eliminate the nested file.

**Cannot open list file**
The destination disk has insufficient space for the listing, or the root directory is full.

**Cannot open response file: *filename***
LINK cannot find the specified response file.

**Cannot open run file**
The destination disk has insufficient space for the .EXE file, or the root directory is full.

**Cannot open temporary file**
The destination disk has insufficient space for the temporary file, or the root directory is full.

**Cannot reopen list file**
The original disk was not replaced when requested. Restart LINK.

**Common area longer than 65536 bytes**
The program has more than 64 KB of communal variables. This error occurs only with programs produced by the Microsoft C Compiler or other compilers that support communal variables.

**Data record too large**
An LEDATA record (in an object module) contains more than 1024 bytes of data. This is a symptom of an error in the compiler used to generate the object module. Document the circumstances and contact Microsoft Corporation.

**Dup record too large**
An LIDATA record (in an object module) contains more than 512 bytes of data. This error may be caused by a complex structure definition or by a series of deeply nested DUP operators.

**File not suitable for /EXEPACK, relink without**
The file linked with the /E switch would have been smaller if it had not been compressed. Relink without the /E switch.

**Fixup overflow near *number* in segment *name* in *filename* offset *number***
A group is larger than 64 KB, the original source file contains an intersegment short jump or intersegment short call, the name of a data item conflicts with that of a library subroutine, or an EXTRN declaration is placed inside the wrong segment.

**Incorrect DOS version, use DOS 2.0 or later**
LINK uses the extended file management calls to provide path support and, thus, does not work with versions of MS-DOS earlier than 2.0.

**Insufficient stack space**
Not enough memory is available to run LINK.

**Interrupt number exceeds 255**
The number specified in the /O:*n* switch is not in the range 0 through 255.

**Invalid numeric switch specification**
An incorrect value was entered with one of the LINK options.

**Invalid object module**
One of the object modules is invalid. Recompile the source file. If the error persists after recompiling, document the circumstances and contact Microsoft Corporation.

**NEAR/HUGE conflict**
Conflicting NEAR and HUGE definitions were given for a communal variable. This error occurs only with programs produced by the Microsoft C Compiler or other compilers that support communal variables.

**Nested left parentheses**
An opening (left) parenthesis is needed on the left side of an overlay module.

**Nested right parentheses**
A closing (right) parenthesis is needed on the right side of an overlay module.

**No object modules specified**
No object file names were specified in the command line or response file.

**Object not found**
One of the object files specified in the command line was not found.

**Out of space on list file**
The destination disk has insufficient space for the listing.

**Out of space on run file**
The destination disk has insufficient space for the .EXE file.

**Out of space on scratch file**
The disk in the default drive has insufficient space for temporary files.

**Overlay manager symbol already defined: *name***
A symbol name was defined that conflicts with one of the special overlay manager names. Use another symbol name.

**Please replace original diskette in drive *X* and press <ENTER>**
The /P switch was specified in the command line and the disk to receive the .EXE file produced by LINK has already been inserted. This message indicates that the .EXE file was successfully created and that the original disk should again be placed in the drive.

**Relocation table overflow**
More than 32768 long calls, long jumps, or other long pointers were found in the program. The program may need to be restructured to reduce the number of FAR references. (Pascal and FORTRAN users should try turning off the debugging option before restructuring the program.)

**Response line too long**
A line in a response file had more than 127 characters.

**Segment limit set too high**
The number specified in the /SE:*n* switch was not in the range 1 through 1024.

**Segment limit too high**
Not enough memory is available for LINK to allocate tables to describe the number of segments requested (default = 128 or the number specified in the /SE:*n* switch). Use the /SE:*n* switch to specify a smaller number of segments, or alter the system configuration to increase the amount of free memory.

**Segment size exceeds 64K**
The program is a small-model program with more than 64 KB of code or data, a compact-model program with more than 64 KB of code, or a medium-model program with more than 64 KB of data. Selection of a different model or alteration of the program code may be required to successfully complete the LINK process.

**Stack size exceeds 65536 bytes**
The size specified for the stack in the /ST:*n* switch was too large, or the combined length of multiple declared stack segments exceeded 64 KB.

**Symbol already defined: "*symbol*"**
One of the special overlay symbols required for overlay support was previously defined.

**Symbol defined more than once: "*symbol*" in file**
A symbol has been defined more than once in the object module. Remove the extra symbol definition.

**Symbol table overflow**
The program has more than 256 KB of symbolic information (publics, externals, segments, groups, classes, files, and so on). Eliminate as many public symbols as possible, combine modules and/or segments, and recreate the object files.

**Terminated by user**
Ctrl-C or Ctrl-Break was pressed, causing the LINK session to be terminated prematurely.

**Too many external symbols in one module**
An object module contains more than the limit of 1023 external symbols.

**Too many group-, segment-, and class-names in one module**
One of the object modules for the program contains too many group, segment, and class names. The source file for the object module may need to be divided or restructured.

**Too many groups**
The program defines more than nine groups (including DGROUP). Groups must be combined or eliminated.

**Too many GRPDEFs in one module**
LINK encountered more than nine group definitions (GRPDEFs) in a single object module. Reduce the number of GRPDEFs or split the object module.

**Too many libraries**
More than 16 libraries were specified. Combine libraries or use object modules that require fewer libraries.

**Too many overlays**
The program defines more than 63 overlays. Reduce the number of overlays.

**Too many segments**
The program has more than the maximum number of segments as specified by the default of 128 or with the /SE:*n* switch. Use the /SE:*n* switch to specify a greater number of segments.

**Too many segments in one module**
An object module has more than 255 segments. Split the module or combine segments.

**Too many TYPDEFs**
An object module contains too many TYPDEF records (these records describe communal variables). This error occurs only with programs produced with the Microsoft C Compiler or other compilers that support communal variables.

**Unexpected end-of-file on library**
This message may indicate that the disk containing the library in use was removed prematurely.

**Unexpected end-of-file on scratch file**
The disk containing VM.TMP was removed.

**Unmatched left parenthesis**
A syntax error was detected in the specification of an overlay structure. Refer to the language compiler manual for instructions on specifying overlays to LINK.

**Unmatched right parenthesis**
A syntax error was detected in the specification of an overlay structure. Refer to the language compiler manual for instructions on specifying overlays to LINK.

**Unrecognized switch error: *"option"***
An unrecognized character was entered after a forward slash (/) in the command line.

**Unresolved COMDEF; Microsoft internal error**
This is a serious problem. Note the circumstances of the failure and contact Microsoft Corporation.

**Unresolved externals: *list***
A symbol was declared external (EXTRN) in one object module but was not declared PUBLIC in the object module in which it was defined, or a necessary library specification was omitted from the command line or response file.

**VM.TMP is an illegal file name and has been ignored**
VM.TMP was specified as an object file name. If an object file named VM.TMP exists, rename it.

**Warning: load-high disables exepack**
The /H and /E switches cannot be used at the same time.

**Warning: no stack segment**
The program contains no segment with the STACK combine type. This message can be ignored if there is a specific reason for not defining a stack (for example, if the .EXE file will subsequently be converted to a .COM file) or for defining one without the STACK combine type.

**WARNING: Segment longer than reliable size**
Although code segments can be as long as 65536 bytes, code segments longer than 65500 bytes can be unreliable on the Intel 80286 microprocessor. Reduce all code segments to 65500 bytes or less.

**Warning: too many public symbols**
The /M switch was used to request a sorted listing of public symbols in the map file, but there are too many symbols to sort. LINK will produce an unsorted listing instead.

# MAKE

Maintain Programs

## Purpose

Interprets a text file of commands to compare dates of files and carry out other operations on the basis of the comparison. MAKE is customarily used to update the executable version of a program after a change to one or more of its source files. The MAKE utility is supplied with the Microsoft Macro Assembler (MASM), C Compiler, and FORTRAN Compiler. This documentation describes MAKE version 4.05.

## Syntax

MAKE [/D] [/I] [/N] [/S] [*name=value* ...] *filename*

where:

| | |
|---|---|
| *filename* | is an ASCII text file that contains MAKE dependency statements, commands, macro definitions, and inference rules. |
| *name=value* | declares a MAKE macro, associating a specific value with the dummy parameter *name*. |
| /D | displays the last modification date of each file as it is scanned. |
| /I | causes MAKE to ignore exit codes returned by programs called by *filename*. |
| /N | displays but does not execute the commands in *filename*. |
| /S | selects "silent" mode (commands are not displayed as they are executed). |

***Note:*** Switches can be either uppercase or lowercase and can be preceded by a dash (-) instead of a forward slash (/). Versions of MAKE earlier than 4.0 have no switches.

## Description

The MAKE utility allows maintenance of complex programs to be automated. Its basic operation is to compare the dates of files and to carry out, or not carry out, an associated list of commands on the basis of the comparison.

The *filename* parameter specifies an ASCII text file often referred to as a make file. By convention, *filename* is the same as the name of the executable program being maintained, but without an extension. A make file can contain the following types of entries:

- Dependency statements
- Commands
- Macro definitions
- Inference rules
- Comments

The basic form of a make file is a dependency statement followed by one or more valid MS-DOS command lines:

*targetfile*: *dependentfile1* [*dependentfile2*...]
    *command1*
    [*command2*]
    ...

where *targetfile* designates the file that may need updating, *dependentfile* is a source file or files on which *targetfile* depends, and *command1*, *command2*, and so forth are any valid MS-DOS internal commands or external programs. These commands or programs are executed only if the date and time stamps of any dependent file are more recent than those of the target file or if the target file does not exist. Only one target file can be specified. Any number of dependent files can be included; each dependent filename must be separated from the next by at least one space. If too many dependent files are included to fit on a single line, the line can be terminated with a backslash character (\) and the list continued on the next line.

Any number of MS-DOS command lines can follow a dependency statement. The last command line should be followed by a blank line to set it off from the next MAKE entry. It is recommended that each command line include a leading space or tab character for compatibility with future versions of MAKE and existing versions of XENIX MAKE.

A macro definition takes the form

*name=value*

where both *name* and *value* are any string. Whenever *name* is referenced in the make file in the form $(*name*), *name* is replaced by the string *value* before the statement that contains it is evaluated or executed. Macro definitions can be nested, although very complex macro definitions can result in the premature termination of the MAKE process because of lack of memory. If *name* is not defined in the file but is defined in the system environment block by a previous SET command, $(*name*) is replaced by the string following the equal sign (=) in the environment block. If the command line contains a parameter of the form *name=value*, the command line overrides any definition of *name* in the make file or in the environment block. Thus, the precedence for macro definitions with the same *name* is

1. Command line
2. Make file
3. Environment block

MAKE contains several special macros that make it more convenient to form commands:

| Macro | Action |
|---|---|
| $* | Substitutes as the base portion of *targetfile* (the filename without the extension). |
| $@ | Substitutes as the complete *targetfile* name. |
| $** | Substitutes as the complete *dependentfile* list. |

An inference rule specifies a series of commands to be carried out for a matching dependency statement that is not followed by its own list of commands. Inference rules allow a set of commands to be applied to more than one *targetfile: dependentfile* description, eliminating repetition of the same set of commands for several descriptions. An inference rule takes the form

*.dependentextension.targetextension*:
    *command1*
    [*command2*]
    ...

Whenever MAKE finds a dependency statement not followed by any commands, the utility first searches the make file for an inference rule. If MAKE doesn't find an inference rule in the make file, the utility then searches the current drive and directory (or any directories specified with the MS-DOS PATH command) for the tools initialization file (TOOLS.INI) and searches the *[make]* section of TOOLS.INI for an inference rule that matches the extensions of the target file and dependent files in the dependency statement.

A make file can contain any number of comment lines. If a comment is placed where MAKE expects to find a command, the comment must be on a separate line and must have the pound character (#) as the first character of the line. Elsewhere, a pound character and following comment text can be placed either on a line alone or after the last dependent file or command listed on a line. Characters appearing on a line after the pound character are ignored during execution.

The /D, /N, and /S switches affect MAKE's output to the display while MAKE is executing. The /D switch causes the last modification date of each file to be displayed as the file is scanned. The /N switch causes the commands in the make file to be expanded and displayed, but not executed; this is useful for determining the result of a specific MAKE process without first examining the file dates and without recompiling or relinking files. The /S switch selects "silent" mode, in which commands are not displayed as they are executed.

The /I switch causes MAKE to ignore error codes returned by the compilers, assemblers, linkers, or other programs called by the make file. When the /I switch is used, the MAKE process proceeds to completion regardless of errors instead of terminating immediately as it ordinarily would, but the resulting files may not be executable.

## Return Codes

0 No error; the MAKE process was successful.
1 Processing was terminated because of a fatal error by MAKE or by one of the programs called by MAKE.

## Example

Assume that the file SHELL contains the following MAKE dependency statements and commands:

```
video.obj: video.asm
        masm video;

shell.obj: shell.c
        msc shell;

shell.exe: shell.obj video.obj
        link /map shell+video,shell,shell,slibc2
```

The SHELL file asserts that the executable program SHELL.EXE is composed of the files SHELL.OBJ and VIDEO.OBJ, which are in turn compiled or assembled from the source files SHELL.C and VIDEO.ASM. To update the file SHELL.EXE if either of the source files for its constituent modules has been changed, type

```
C>MAKE SHELL  <Enter>
```

## Messages

**fatal error U1001: macro definition larger than 512**
A single macro was defined to have a value string longer than the 512-byte maximum. Rewrite the make file to use two or more short lines instead of one long line.

**fatal error U1002: infinitely recursive macro**
The macros defined in the make file form a circular chain.

**fatal error U1003: out of memory**
The make file cannot be processed because insufficent memory is available in the transient program area. Split the make file into two make files or reconfigure the system to increase available memory.

**fatal error U1004: syntax error : macro name missing**
A macro name is missing from the left side of the equal sign (=).

**fatal error U1005: syntax error : colon missing**
A line that should be a dependency statement lacks the colon that separates a target file from its dependent files. MAKE expects any line that follows a blank line to be a dependency statement.

**fatal error U1006: *targetname* : macro expansion larger than 512**
A single macro expansion, plus the length of any string to which it may be concatenated, is longer than 512 bytes. Rewrite the make file to use two or more short lines instead of one long line.

**fatal error U1007: multiple sources**
An inference rule has been defined more than once in the make file.

**fatal error U1008: *filename* : cannot find file**
The specified file does not exist.

**fatal error U1009: *command* : argument list too long**
A command line in the make file is longer than 128 characters (the maximum MS-DOS allows).

**fatal error U1010: *filename* : permission denied**
The specified file is read-only.

**fatal error U1011: not enough memory**
Memory is insufficient in the transient program area to execute a program listed in the make file. Reconfigure the system to increase available memory, if necessary.

**fatal error U1012: *filename* : unknown error**
This is a serious problem. Note the circumstances of the failure and notify Microsoft Corporation.

**fatal error U1013: *command* : error *returncode***
One of the programs or commands called by MAKE was not able to execute correctly. MAKE terminates and displays the error code from the program that failed.

**warning U4000: *filename* : target does not exist**
The target file does not already exist. The dependency statement is evaluated as though the target file exists and has a date earlier than that of any of the dependent files.

**warning U4001: dependent *filename* does not exist;**
**target *filename* not built**
One of the dependent files does not exist or could not be found, so MAKE terminated without creating a new target file.

**warning U4013: *command* : error *returncode* (ignored)**
One of the programs or commands called by MAKE did not execute successfully and has returned the specified return code. Because MAKE was run with the /I switch, MAKE ignores the error and continues processing the make file.

**warning U4014: usage : make [/n] [/d] [/i] [/s] [name=value ...] file**
An error was detected in the MAKE command line.

# MAPSYM

Create Symbol File for SYMDEB

## Purpose

Processes a map file generated by the Microsoft Object Linker (LINK) to create a special symbol file for use with SYMDEB, the symbolic debugging program. The MAPSYM utility is supplied with the Microsoft Macro Assembler (MASM). This documentation describes MAPSYM version 4.0.

## Syntax

MAPSYM [/L] *map_file*

where:

| | |
|---|---|
| *map_file* | is a map file produced by LINK (default extension = .MAP). |
| /L | causes information about the symbol file to be displayed as it is created. |

***Note:*** The /L switch can be either uppercase or lowercase and can be preceded by a dash (-) instead of a forward slash (/).

## Description

LINK combines relocatable object records (produced by MASM or a high-level-language compiler) into an executable program, which is stored in a specially formatted file with a .EXE extension. LINK can also produce an optional map file that contains information about public symbols and addresses in the linked program. The map file is an ordinary ASCII text file and has a default extension of .MAP.

To create a map file to use with MAPSYM, the LINK command line should include the /MAP switch, which creates the file, and the /LINENUMBERS switch, which includes line numbers. *See* PROGRAMMING UTILITIES: LINK.

The MAPSYM utility processes a map file into a special symbol file that can be used by SYMDEB. A drive and pathname can be specified if the map file is not in the current directory. If a file extension is not specified, .MAP is assumed.

The symbol file created by MAPSYM is placed in the current directory and has the same name as the map file but has the extension .SYM. It can contain a maximum of 1024 segments (or as many segments as can fit into available memory) and 10,000 symbols per segment. *See* PROGRAMMING UTILITIES: SYMDEB.

When the /L switch precedes *map_file* in the command line, MAPSYM displays the names of groups defined in the program described by the map and symbol files, plus the program's starting address. The /L switch does not affect the format of the symbol file that is generated.

## Return Codes

0 No error; the MAPSYM process was successful.
1 Processing was terminated because of a write failure, because the map file specified does not exist, or because the symbol file could not be created.
4 Processing was terminated because an unexpected end-of-file mark was detected, because too many segments exist in the map file, because no public symbols exist in the map file, or because not enough memory is available to create the symbol file.

## Example

To convert the file HELLO.MAP, which was produced by assembling and linking the file HELLO.ASM, to a symbol file that can be used by SYMDEB, type

```
C>MAPSYM /L HELLO  <Enter>
```

MAPSYM displays the following:

```
Microsoft (R) Symbol File Generator  Version 4.00
Copyright (C) Microsoft Corp 1984, 1985.  All rights reserved.
Building: HELLO.SYM
HELLO.MAP
        Program entry point at 0000:0100
HELLO      0 segment
```

The symbol file produced by MAPSYM symbol has the name HELLO.SYM.

## Messages

**Can't create: <*filename*>**
The drive specified does not exist, the current disk or directory is full, or the output file already exists and is read-only.

**Can't open MAP file: <*filename*>**
The file named in the command line does not exist.

**DOS 2.0 or later required**
MAPSYM does not work with versions of MS-DOS earlier than 2.0.

**mapsym: out of memory**
System memory is insufficient to process the map file.

**mapsym: segment table (*n*) exceeded.**
More than 1024 segments have been used in the map file. The number displayed is the total number of segments in the map file.

**No public symbols**
**Re-link file with the /M switch!**
The map file created by LINK does not include a list of public names. The .EXE file must be relinked using the /MAP switch to generate a map file that can be used with MAPSYM.

**Unexpected eof reading: <*filename*>**
The map file contains no symbols, is corrupt, or is otherwise invalid. The .EXE file must be relinked and a new map file generated.

**usage: MAPSYM [/l] maplist**
A syntax error was detected in the command line.

**Write fail on: <*filename*>**
An error occurred during the creation of the output file.

# MASM

Microsoft Macro Assembler

## Purpose

Translates an assembly-language source program into a relocatable object module. MASM is part of the Microsoft Macro Assembler (MASM) retail package. This documentation describes MASM version 4.0.

## Syntax

MASM

or

MASM *source_file* [,[*object_file*][,[*list_file*][,[*cref_file*]]]] [*options*] [;]

where:

*source_file* is the name of the file containing the assembly-language source code (default extension = .ASM).

*object_file* is the name of the file to receive the assembled object module (default extension = .OBJ).

*list_file* is the name of the file or device to receive the assembly listing (default = NUL). (If destination = file, default extension = .LST.)

*cref_file* is the name of the cross-reference file to receive information for later processing by the CREF utility (default = NUL). (If destination = file, default extension = .CRF.)

*options* is one or more switches from the list below.

| | |
|---|---|
| /A | Writes the program segments in alphabetic order. |
| /B*n* | Sets the size of the source-file buffer in kilobytes (1–63, default = 32). |
| /C | Creates a cross-reference (.CRF) file. |
| /D | Adds a first-pass program listing to *list_file* if a list file was specified (default = second-pass listing only). |
| /D*symbol* | Defines *symbol* as a null text string. |
| /E | Assembles code for an 8087/80287 emulator. |
| /I*path* | Defines a directory to be searched for *include* files. |
| /L | Creates a list (.LST) file with line-number information. |
| /ML | Preserves case sensitivity in all symbol names. |

*(more)*

| | |
|---|---|
| /MU | Converts all lowercase names to uppercase names. |
| /MX | Preserves lowercase in public and external names only. |
| /N | Suppresses generation of tables of macros, structures, records, groups, segments, and symbols at the end of the list file. |
| /P | Checks for impure code in 80286 protected mode; has no effect unless the .286P directive is included in the source file. |
| /R | Assembles code for an 8087/80287 math coprocessor. |
| /S | Arranges program segments in order of occurrence. |
| /T | Selects terse mode, suppressing all messages generated during assembly except error messages. |
| /V | Selects verbose mode, displaying the number of lines and symbols at the end of assembly. |
| /X | Includes false conditionals in the list file. |
| /Z | Displays source lines with errors during assembly. |

***Note:*** Switches can be either uppercase or lowercase and can be preceded by a dash (-) instead of a forward slash (/).

## Description

MASM translates assembly-language source code into relocatable object modules. The object modules can then be placed in a library file or processed by the Microsoft Object Linker (LINK) to create an executable program.

The *source_file* parameter is the only required filename. It specifies a file containing the assembly-language source code in ASCII text. If no extension is specified, MASM uses .ASM. If no source file is entered in the command line, MASM prompts for a source file name.

The *object_file* parameter specifies the file that will contain the assembled relocatable object code. If this parameter is not supplied, MASM uses the same filename as *source_file* but substitutes the extension .OBJ.

The *list_file* parameter specifies a destination file or device for the optional program listing. The listing contains the original source code, the assembled machine code, macro definitions and expansions, and other useful information, formatted into pages with titles, dates, and page numbers. If the destination of the listing is a file, the file's default extension is .LST. If the *list_file* parameter is not included in the command line, MASM sends the listing to NUL (that is, a listing is not produced).

The *cref_file* parameter specifies the name of a cross-reference file to receive information to be processed by the CREF utility. If a file extension is not specified, MASM uses .CRF. If the *cref_file* parameter is not included in the command line, MASM sends the file to NUL (that is, no cross-reference file is generated).

If the MASM command is entered without parameters, MASM prompts the user for each filename. The default response for each prompt (except the source file prompt) is displayed in square brackets and can be selected by pressing the Enter key.

After the source file is specified, if MASM encounters a semicolon character (;) in the command line or at any prompt, it uses default values for the remaining parameters. MASM ignores any parameters specified after the semicolon.

MASM does two passes to translate the assembly-language code in the source file into relocatable object code. Any errors detected during translation are displayed on standard output and included in the program listing (if one is requested). Two types of errors may be detected: warning errors and severe errors. If MASM encounters a warning error, it still creates the object file, although the resulting file may be unusable. If MASM encounters a severe error, it does not create the object file. After a file has been successfully assembled without errors, the LINK utility can be used to convert the resulting object file into an executable program file.

MASM supports a wide variety of options that can be selected by including switches in the command line or by responding to any prompt.

The /A and /S switches determine the order of segments in the resulting object module file. The /A switch places the segments into the object file in alphabetic order. The /S switch (the default) arranges the segments in the same order they occur in the source file.

The /B*n*, /D*symbol*, and /I*path* switches have rather general effects on the behavior of MASM. The /B*n* switch sets the size (in kilobytes) of the source file's RAM buffer; the value of *n* must be between 1 and 63, inclusive (default = 32). If the RAM buffer is large enough, the entire source file can be kept resident in memory, reducing disk activity during passes. The /D*symbol* switch defines a null text-string symbol from the command line. This symbol can be referenced inside the program with the IFDEF directive to control the conditional assembly of portions of the program. The /I*path* switch specifies a directory that will be searched for files named in assembler INCLUDE statements if those statements do not include an explicit directory. As many as 10 such search paths can be specified with individual /I*path* switches.

The /E and /R switches affect the generation of code for the 8087/80287 emulator or 8087/80287 math coprocessor. (Support for the 80287 is included with MASM versions 3.0 and later.) The /E switch generates software interrupts to floating-point-processor emulator routines. A subprogram assembled with the /E switch can be linked to C, Pascal, and FORTRAN programs and can use the emulator libraries. The /R switch produces in-line machine instructions for the math coprocessor when floating-point mnemonics are used.

The /ML, /MU, and /MX switches control MASM's handling of uppercase and lowercase names. The /ML switch makes MASM case sensitive; that is, it makes MASM differentiate a

name in uppercase letters from the same name in lowercase letters. (The /ML switch should not be used if the source file contains 8087 WAIT instructions and MASM 4.0 is being used to translate the file.) The /MU switch (the default) makes MASM case insensitive; all lowercase letters are converted to uppercase for purposes of assembly. The /MX switch makes MASM case sensitive for public and external names only (names defined with PUBLIC or EXTRN directives). The /MX switch is often used to process assembly-language functions for C programs.

The /P switch checks for impure code segments that will cause problems if the assembled program is run in 80286 protected mode. The switch checks by flagging any instruction that will change a memory location addressed through the processor's CS register. The /P switch has no effect unless the assembly-language source file includes the .286P directive.

The /C, /D, /L, /N, and /X switches control the contents of the program listing and other optional files that are generated as a result of assembly. The /C switch causes the creation of a cross-reference (.CRF) file and the addition of line numbers to the list (.LST) file (if one exists). The /C switch should be included in the command line if the cross-reference file will be used later with the CREF utility to produce a cross-reference listing. The /D switch includes a listing from the first pass as well as a listing from the second pass in the list file if a list file was specified (default = second-pass listing only). By comparing the two listings, the user can isolate an instruction causing a phase error. (A phase error occurs when MASM makes assumptions about addresses, values, or data types on the first pass that are not valid in the second pass.) The /L switch creates a list file with line-number information and gives it the same name as the source file, with the extension .LST. The /N switch suppresses generation of tables — symbols, segments, groups, structures, records, and macros — at the end of a program listing. The /X switch includes statements inside false conditional statements in the list file, allowing conditionals that do not generate code to be displayed. /X has no effect if the .SFCOND or the .LFCOND directive is used in the source file; if the .TFCOND directive is used, the effects of /X are reversed.

***Note:*** The effects of /X are also reversed in MASM version 1.2. In that version, statements within a false conditional are included in the list file by default, and /X will suppress them.

The /T, /V, and /Z switches affect MASM's display on standard output. The /T (terse) switch suppresses messages to standard output, except for messages indicating warning errors or severe errors. The /V (verbose) switch displays information about the number of source lines and symbols at the end of the assembly, in addition to displaying the normal error and symbol space information. The /Z switch displays the actual source lines producing assembly errors (rather than displaying just the error type and line number).

***Note:*** Versions of MASM earlier than 4.0 always show both the source line and the error message.

## Return Codes

0 No errors were found during assembly.
1 An error was detected in one of the command-line parameters.
2 The assembly-language source file could not be opened.
3 The list file could not be created.
4 The object file could not be created.
5 The cross-reference file could not be created.
6 An *include* file could not be opened.
7 At least one severe error was detected during assembly. (MASM deletes the invalid object file.)
8 The assembly was terminated because a memory allocation error occurred.
10 An error occurred in defining a symbol (with the /D*symbol* switch) from the command line.
11 Assembly was interrupted by the user's pressing Ctrl-C or Ctrl-Break.

## Examples

To assemble the source file CLEAN.ASM in the current drive and directory and place the resulting relocatable object module in the file CLEAN.OBJ without producing a listing or a cross-reference file, type

```
C>MASM CLEAN;  <Enter>
```

The semicolon after the first parameter causes MASM to use the default values for the rest of the parameters.

To assemble the source file CLEAN.ASM, put the object code in a file named CLEAN.OBJ, create a list file named CLEAN.LST, and place information for later processing by the CREF utility in the cross-reference file CLEAN.CRF, type

```
C>MASM CLEAN,CLEAN,CLEAN,CLEAN  <Enter>
```

or

```
C>MASM CLEAN,,CLEAN,CLEAN  <Enter>
```

To use MASM interactively, enter its name without parameters:

```
C>MASM  <Enter>
```

MASM then prompts for all the necessary information. For example, the interactive session on the next page assembles the file HELLO.ASM into the file HELLO.OBJ, producing no listing or .CRF file.

```
C>MASM  <Enter>
Microsoft (R) Macro Assembler  Version 4.00
Copyright (C) Microsoft Corp 1981, 1983, 1984, 1985.  All rights reserved.

Source filename: [.ASM]: HELLO  <Enter>
Object filename: [HELLO.OBJ]:  <Enter>
Source listing [NUL.LST]:  <Enter>
Cross-reference [NUL.CRF]:  <Enter>

  51004 Bytes symbol space free

    0 Warning Errors
    0 Severe Errors
```

## Messages

**8087 opcode can't be emulated**
An 8087 opcode or the operands used with it produced an instruction the emulator cannot support.

**Already defined locally**
An attempt was made to define a symbol as EXTRN that had already been defined locally.

**Already had ELSE clause**
An attempt was made to define an ELSE clause within an existing ELSE clause. (ELSE cannot be nested without nesting IF...ENDIF.)

**Already have base register**
More than one base register was specified within an operand.

**Already have index register**
More than one index register was specified within an operand.

**Block nesting error**
A segment, structure, macro, IRC, IRP, REPT, or nested procedure was not terminated properly.

**Byte register is illegal**
A byte register was used incorrectly in an instruction.

**Can't override ES segment**
An attempt was made to override the ES segment in an instruction in which this override is invalid.

**Can't reach with segment reg**
No ASSUME directive was given to make the variable reachable.

**Can't use EVEN on BYTE segment**
An EVEN directive was used on a segment declared to be a byte segment.

**Circular chain of EQU aliases**
An alias EQU ultimately points to itself.

**Constant was expected**
A constant was expected, but an item was received that does not evaluate to a constant.

**CS register illegal usage**
The CS register was used incorrectly in one of the instructions.

**Data emitted with no segment**
Code that is not located within a segment attempted to generate data.

**Directive illegal in STRUC**
All statements within STRUC blocks must be either comments preceded by a semicolon character (;) or one of the define directives (DB, DW, and so on).

**Division by 0 or overflow**
An expression was encountered that resulted in either a division by 0 or a number too large to be represented.

**DUP is too large for linker**
Nesting of DUP operators was such that a record too large for LINK was created.

**End of file, no END directive**
No END statement was encountered, or a nesting error occurred.

**Extra characters on line**
Superfluous characters were detected on a line after sufficient information to define an instruction was interpreted.

**extra file name ignored**
The command line contained more than four filename parameters.

**Field cannot be overridden**
An attempt was made to give a value to a field that cannot be overridden with a STRUC initialization statement.

**Forced error**
An error was forced with the .ERR directive.

**Forced error - expression equals 0**
An error was forced with the .ERRE directive.

**Forced error - expression not equal 0**
An error was forced with the .ERRNZ directive.

**Forced error - pass1**
An error was forced with the .ERR1 directive.

**Forced error - pass2**
An error was forced with the .ERR2 directive.

**Forced error - string blank**
An error was forced with the .ERRB directive.

**Forced error - string not blank**
An error was forced with the .ERRNB directive.

**Forced error - strings different**
An error was forced with the .ERRDIF directive.

**Forced error - strings identical**
An error was forced with the .ERRIDN directive.

**Forced error - symbol defined**
An error was forced with the .ERRDEF directive.

**Forced error - symbol not defined**
An error was forced with the .ERRNDEF directive.

**Forward reference is illegal**
An item was referenced in the operand of an EQU or equal-sign (=) directive before it was defined.

**Illegal register value**
A specified register value does not fit into the *reg* field (that is, the value is greater than 7).

**Illegal size for item**
The size of the referenced item is invalid. This error also frequently occurs when an attempt is made to assemble source code written for assemblers with less strict type-checking than that of the Microsoft Macro Assembler (such as early versions of the IBM assembler). The problem can usually be solved by overriding the type of the operand with the PTR operator.

**Illegal use of external**
A variable that was declared external was used incorrectly.

**Illegal use of register**
An attempt was made to use a register with an instruction in which a register cannot be used.

**Illegal value for DUP count**
The DUP count was not a constant that evaluates to a positive integer greater than zero.

**Improper operand type**
An operand was used in a way that prevents opcode generation.

**Improper use of segment register**
An attempt was made to use a segment register in an instruction in which use of a segment register is not permitted.

**Impure memory reference**
An attempt was made to store data in the code segment when the .286P directive and the /P switch were in effect.

**Index displ. must be constant**
An index displacement was used incorrectly or did not evaluate to an absolute number or memory address.

**Internal error**
An internal logic error was detected in the assembler. Document the circumstances and contact Microsoft Corporation.

**Label can't have seg. override**
A segment override was used incorrectly.

**Left operand must have segment**
The content of the right operand requires that a segment be specified in the left operand.

**Line too long expanding *symbol***
A symbol defined by an EQU or equal-sign (=) directive is so long that expanding it will cause the assembler's internal buffers to overflow. This message may indicate a recursive text macro.

**Missing data; zero assumed**
An operand is missing from a statement and MASM assumes its value is zero. This is a warning error; the object file is not deleted as it is with severe errors.

**More values than defined with**
Too many initial values were given when defining a variable using a REC or STRUC type.

**Must be associated with code**
A data-related item was used where a code-related item was expected.

**Must be associated with data**
A code-related item was used where a data-related item was expected.

**Must be AX or AL**
A register other than AX or AL was specified where only these are acceptable.

**Must be in segment block**
An attempt was made to generate code by instructions that were not contained within a segment.

**Must be index or base register**
An instruction requires a base or index register, and some other register was specified within square brackets ([]).

**Must be record field name**
A record field name was expected, but something else was encountered.

**Must be record or fieldname**
A record name or field name was expected, but something else was encountered.

**Must be register**
A register was expected as the operand, but something else was encountered.

**Must be segment or group**
A segment or group was expected, but something else was encountered.

**Must be structure field name**
A structure field name was expected, but something else was encountered.

**Must be symbol type**
A BYTE, WORD, DWORD, or similar designation was expected, but something else was encountered.

**Must be var, label or constant**
A variable, label, or constant was expected, but something else was encountered.

**Must have opcode after prefix**
A REP, REPE, REPNE, REPZ, or REPNZ instruction was not followed by the mnemonic for a string operation.

**Near JMP/CALL to different CS**
An attempt was made to do a NEAR jump or call to a location in a code segment defined with a different ASSUME:CS.

**No immediate mode**
Immediate data was supplied as an operand for an instruction that cannot use immediate data. For example, immediate data cannot be moved directly with a MOV instruction to a segment register; it must first be moved into a general register and then copied to the segment register.

**No or unreachable CS**
An attempt was made to jump to a label that is unreachable.

**Normal type operand expected**
A STRUC, BYTE, WORD, or some other invalid operand was encountered when a variable label was expected.

**Not in conditional block**
An ENDIF or ELSE statement was encountered, and no previous conditional-assembly directive was active.

**Not proper align/combine type**
The SEGMENT parameters are incorrect. Check the align and combine types to be sure they are valid.

**One operand must be const**
The addition operator was used incorrectly.

**Only initialize list legal**
An attempt was made to use a STRUC name without angle brackets (<>).

**Operand combination illegal**
A two-operand instruction was specified and the combination specified was invalid.

**Operand must have segment**
A SEG directive was used incorrectly.

**Operand must have size**
An operand was encountered that needed a specified size, but none had been provided. Often this error can be remedied by using the PTR operator to specify a size type.

**Operand not in IP segment**
An operand cannot be accessed because it is not in the segment last assigned to CS with an ASSUME directive.

**Operand types must match**
MASM encountered different kinds or sizes of arguments in a case where they must match.

**Operand was expected**
MASM expected an operand, but an operator was encountered.

**Operands must be same or 1 abs**
The subtraction operator was used incorrectly.

**Operator was expected**
MASM expected an operator, but an operand was encountered.

**Out of memory**
System memory is insufficient to complete the assembly. If a listing (.LST) or cross-reference (.CRF) file was being generated, retry the assembly, generating only an object file. It may also be necessary to modify the source program to reduce the load on the symbol table (by shortening names or reducing the number of EQU statements or macros, for example).

**Override is of wrong type**
An attempt was made to use a data item of incorrect size in a STRUC initialization statement.

**Override value is wrong length**
The override value for a structure field is too large to fit in the field.

**Override with DUP is illegal**
An attempt was made to use DUP to override in a STRUC initialization statement.

**Phase error between passes**
The program has ambiguous instruction directives that caused the location of a label in the program to change in value between the first and second passes of MASM. A common cause is a forward reference to a typed data item in the instructions preceding the label that generated the phase error message. Use the /D switch to produce a first-pass listing to aid in resolving phase errors between passes.

**Redefinition of symbol**
This message is displayed during first pass upon the second declaration of a symbol that has been defined in more than one place.

**Reference to mult defined**
The instruction references a symbol that has been defined more than once.

**Register already defined**
An internal error was detected. Note the circumstances of the failure and contact Microsoft Corporation.

**Relative jump out of range**
A conditional jump references a label that is out of the allowed range of –128 to +127 bytes relative to the current instruction. The problem usually can be corrected by reversing the condition of the jump and using an unconditional jump (JMP) to the out-of-range label.

**Segment parameters are changed**
The list of parameters encountered for a SEGMENT was not identical to the list specified the first time the segment was used.

**Shift count is negative**
A shift expression was generated that resulted in a negative shift count.

**Should have been group name**
A group name was expected, but something else was encountered.

**Symbol already different kind**
An attempt was made to redefine an already defined symbol.

**Symbol has no segment**
An attempt was made to use a variable with SEG that has no known segment.

**Symbol is already external**
An attempt was made to redefine a symbol as local that has already been defined as external.

**Symbol is multi-defined**
This message is displayed during the second pass upon each declaration of a symbol that has been defined in more than one place.

**Symbol is reserved word**
An attempt was made to use a reserved MASM word as a symbol.

**Symbol not defined**
A symbol that had not been defined was used.

**Symbol type usage illegal**
A PUBLIC symbol was used incorrectly.

**Syntax error**
The syntax of the statement does not match any recognizable syntax.

**Type illegal in context**
The type specified is of an unacceptable size.

**Unable to open input file *filename***
The specified source file cannot be found.

**unknown switch *letter***
The command line included an invalid switch.

**Unknown symbol type**
MASM does not recognize the size type specified in a label or external declaration. Rewrite with a valid type such as BYTE, WORD, or NEAR.

**Value is out of range**
A value is too large for its expected use.

**Wrong type of register**
A directive or instruction expected one type of register, but another type was encountered.

# DEBUG

Program Debugger

## Purpose

Allows the controlled execution of a program for debugging purposes or the alteration of the binary contents of any file. The DEBUG utility is supplied with the MS-DOS distribution disks.

## Syntax

DEBUG

or

DEBUG *filename* [*parameter*...]

where:

| | |
|---|---|
| *filename* | is the name of the file that contains data to be modified or a program to be debugged. If *filename* includes an extension, it must be specified. |
| *parameter*... | is one or more filenames or switches required by a program being debugged. |

## Description

The DEBUG program allows a file to be loaded, examined, and altered. If the file is not a .EXE file or a .HEX file, it may also be written back to disk. If the file contains a program, the program can be disassembled, modified, traced one instruction at a time, or executed at full speed with preset breakpoints. DEBUG can also be used to read from and write to input/output (I/O) ports and to read, modify, and write absolute disk sectors.

The command line typically includes the *filename* parameter, which is the name of an executable program (with the extension .COM or .EXE) to be loaded into DEBUG's memory buffer. Files with the extension .EXE are loaded in a manner compatible with the MS-DOS loader; if necessary, the contents of the file are relocated so that the program is ready to execute. Files with the extension .HEX are converted to binary images and loaded at the internally specified address. All other files are assumed to be direct memory images and are read directly into memory starting at offset 100H.

An appropriate program segment prefix (PSP) is synthesized at the head of DEBUG's buffer for use by the target program (the program being debugged). The PSP includes a command tail at offset 80H and default file control blocks (FCBs) at offsets 5CH and 6CH, constructed from the optional parameters following *filename*.

After DEBUG is loaded and the first file named in the command line is also located and loaded, DEBUG displays its special prompt character, a hyphen (-), and awaits a command. DEBUG commands consist of a single letter, usually followed by one or more

parameters. Uppercase and lowercase characters are treated the same except when they are contained in strings enclosed within single or double quotation marks. All commands are executed by pressing the Enter key.

The DEBUG commands are

| Command | Action |
|---|---|
| A | Assemble machine instructions (versions 2.0 and later). |
| C | Compare memory areas. |
| D | Display memory. |
| E | Enter data. |
| F | Fill memory. |
| G | Go execute program. |
| H | Perform hexadecimal arithmetic. |
| I | Input from port. |
| L | Load file or sectors. |
| M | Move (copy) data. |
| N | Name file or command-tail parameters. |
| O | Output to port. |
| P | Proceed through loop or subroutine (versions 3.0 and later). |
| Q | Quit debugger. |
| R | Display or modify registers. |
| S | Search memory. |
| T | Trace program execution. |
| U | Disassemble (unassemble) program. |
| W | Write file or sectors. |

The parameters for a DEBUG command include addresses, ranges, 8-bit or 16-bit hexadecimal values, and lists. Multiple parameters can be separated by spaces, tabs, or commas, but separators are *required* only between hexadecimal values.

An address can be a simple offset or a complete address in the form *segment:offset*. The offset is always a hexadecimal number in the range 00H through FFFFH; the segment can be either a hexadecimal value in the same range or a two-character segment register name (CS, DS, ES, or SS). If the segment portion of an address is absent, DEBUG uses DS unless an A, G, L, T, U, or W command is used, in which case DEBUG uses CS.

A range specifies an area of memory and can be expressed as either two addresses or a starting address and a length. A segment can be included only in the first element of a range; an error message is displayed if a segment is found in the second address. A length is represented by the letter L, followed by a hexadecimal value between 00H and FFFFH that indicates the number of bytes following the starting address that the command should operate on.

***Note:*** Any length that causes an address to exceed 16 bits will generate an error.

A byte, or 8-bit, value is entered as one or two hexadecimal digits, whereas a word, or 16-bit, value is entered as one to four hexadecimal digits. Leading zeros can be omitted.

A list is composed of one or more byte values or strings, separated by spaces, commas, or tabs. A string is one or more ASCII characters enclosed within single or double quotation marks. Case is significant within a string. If the same type of quote character that is used to delimit the string occurs inside the string itself, the character must be doubled inside the string in order to be interpreted correctly. For example:

```
"This ""string"" is OK."
```

When used, a list must be the last parameter in the command line.

DEBUG responds to an invalid command by pointing to the approximate location of the error with a caret character (^) and displaying the word *Error*. For example:

```
-D CS:0100,CS:0200  <Enter>
             ^ Error
```

DEBUG maintains a set of virtual CPU registers for a program being debugged. These registers can be examined and modified with DEBUG commands. When a program is first loaded for debugging, the virtual registers are initialized with the following values:

| Register | .COM Program | .EXE Program |
|---|---|---|
| AX | Valid drive error code | Valid drive error code |
| BX | Upper half of program size | Upper half of program size |
| CX | Lower half of program size | Lower half of program size |
| DX | Zero | Zero |
| SI | Zero | Zero |
| DI | Zero | Zero |
| BP | Zero | Zero |
| SP | FFFEH or top of available memory minus 2 | Size of stack segment |
| IP | 100H | Offset of entry point within target program's code segment |
| CS | PSP | Base of target program's code segment |
| DS | PSP | PSP |
| ES | PSP | PSP |
| SS | PSP | Base of target program's stack segment |

***Note:*** DEBUG checks the first three parameters in the command line. If the second and third parameters are filenames, DEBUG checks any drive specifications with those filenames to verify that they designate valid drives. Register AX contains one of the following codes:

| Code | Meaning |
|---|---|
| 0000H | The drives specified with the second and third filenames are both valid, or only one filename was specified in the command line. |
| 00FFH | The drive specified with the second filename is invalid. |
| FF00H | The drive specified with the third filename is invalid. |
| FFFFH | The drives specified with the second and third filenames are both invalid. |

DEBUG also maintains a set of virtual flags, which may be set or cleared. The flags are

| Flag Name | Value If Set (1) | Value If Clear (0) |
|---|---|---|
| Overflow | OV (Overflow) | NV (No Overflow) |
| Direction | DN (Down) | UP (Up) |
| Interrupt | EI (Enabled) | DI (Disabled) |
| Sign | NG (Minus) | PL (Plus) |
| Zero | ZR (Zero) | NZ (Not Zero) |
| Aux Carry | AC (Aux Carry) | NA (No Aux Carry) |
| Parity | PE (Even) | PO (Odd) |
| Carry | CY (Carry) | NC (No Carry) |

Before DEBUG transfers control to the target program, it saves the actual CPU registers and then loads them with the current values of the virtual registers. Conversely, when control reverts to DEBUG from the target program, the returned register contents are stored back in the virtual register set for inspection and alteration by the user.

## Examples

To load the file SHELL.EXE in the current directory for execution under the control of DEBUG, type

```
C>DEBUG SHELL.EXE  <Enter>
```

To use the DEBUG program to inspect or modify memory or to read, modify, and write absolute disk sectors, simply type

```
C>DEBUG  <Enter>
```

## Message

**File not found**
The filename supplied as the first parameter in the DEBUG command line cannot be found.

# DEBUG: A

Assemble Machine Instructions

## Purpose

Allows entry of assembler mnemonics and translates them into executable machine code.

## Syntax

A [*address*]

where:

*address* is the starting location for the assembled machine code.

## Description

The Assemble Machine Instructions (A) command accepts assembly-language statements, rather than hexadecimal values, for the Intel 8086/8088 microprocessors and the Intel 8087 math coprocessor and then assembles each statement into executable machine code.

The *address* parameter specifies the location where entry of assembly-language mnemonics will begin. If *address* is omitted, DEBUG uses the address following the last instruction generated the last time the A command was used. If the A command has not been used, DEBUG uses the current value of the target program's CS:IP registers.

After an A command is entered, DEBUG prompts for each assembly-language statement by displaying the address, in the form of a segment and an offset, in which the assembled code will be stored. When the Enter key is pressed, the assembly-language statement is translated, and each byte of the resulting machine instruction is stored sequentially in memory (overwriting existing information), beginning at the displayed address. The address following the last byte of the machine instruction is then displayed so that the user can enter the next assembly-language statement. Pressing the Enter key alone in response to the address prompt terminates the A command.

The syntax of assembly-language statements accepted by the DEBUG A command differs slightly from that of the usual Microsoft Macro Assembler programming statements. The differences can be summarized as follows:

- All numbers are assumed to be hexadecimal integers and should be entered without a trailing H character.
- Segment overrides must be specified by preceding the entire instruction with CS:, DS:, ES:, or SS:.
- File control directives (NAME, PAGE, TITLE, and so forth), macro definitions, record structures, and conditional assembly directives are not supported by DEBUG.
- Specific hexadecimal values, rather than program labels, must be included.

- When the data type (word or byte) is not implicit in the instruction, the type must be specified by preceding the operand with BYTE PTR (or BY) or WORD PTR (or WO).
- The size of the string in a string operation must be specified by adding a B (byte) or W (word) to the string instruction mnemonic (for example, LODSB or LODSW).
- The DB and DW instructions accept a parameter of the type *list* and assemble byte and word values directly.
- The WAIT or FWAIT opcodes for 8087 assembler statements are not generated by default, so they must be coded explicitly.
- Memory locations are differentiated from immediate operands by enclosing memory addresses in square brackets.
- Repeat prefixes, such as REP, REPZ, or REPNZ, can be entered either alone on the line preceding the statement they affect or immediately preceding the statement on the same line.
- Although the assembler generates the optimal form (SHORT, NEAR, or FAR) for jumps or calls, depending on the destination address, these designations can be overridden by preceding the operand with a NEAR (or NE) or FAR (no abbreviation) prefix.
- The mnemonic for a FAR RETURN is RETF.

## Examples

To begin assembling code at address CS:0100H, type

```
-A 100  <Enter>
```

To assemble the instruction sequence

```
LODS  WORD PTR [SI]
XCHG  BX,AX
JMP   [BX]
```

beginning at address CS:0100H, the following dialogue would take place:

```
-A 100  <Enter>
1983:0100  LODSW  <Enter>
1983:0101  XCHG BX,AX  <Enter>
1983:0103  JMP [BX]  <Enter>
1983:0105  <Enter>
```

To continue assembling at the location following the last instruction generated by a previous A command, type

```
-A  <Enter>
```

# DEBUG: C

Compare Memory Areas

## Purpose

Compares two areas of memory and reports any differences.

## Syntax

C *range address*

where:

| | |
|---|---|
| *range* | is the starting and ending addresses or the starting address and length of the first area of memory to be compared. |
| *address* | is the starting address of the second area of memory to be compared. |

## Description

The Compare Memory Areas (C) command compares the contents of two areas of memory. The location and contents of any differing bytes are displayed in the following format:

*address1 byte1 byte2 address2*

If no differences are found, the DEBUG prompt returns.

The *range* parameter specifies the starting and ending addresses or the starting address and length in bytes of the first area of memory to be compared. The *address* parameter specifies the beginning address of the second area of memory to be compared. If a segment is not included in *range* or *address*, DEBUG uses DS.

## Example

To compare the 64 bytes beginning at CS:CE00H with the 64 bytes beginning at CS:CF0AH, type

```
-C CS:CE00 CE3F CS:CF0A  <Enter>
```

or

```
-C CS:CE00 L40 CS:CF0A  <Enter>
```

If any differences are found, DEBUG displays them in the following format:

```
2124:CE06  00  FF  2124:CF10
```

# DEBUG: D

Display Memory

## Purpose

Displays the contents of an area of memory in hexadecimal and ASCII format.

## Syntax

D [*range*]

where:

*range* is the starting and ending addresses or the starting address and length of the area to be displayed.

## Description

The Display Memory (D), or Dump, command displays the contents of a specified range of memory addresses in hexadecimal and ASCII format.

The *range* parameter gives the starting and ending addresses or the starting address and length in bytes of the memory to be displayed. If *range* does not include a segment, DEBUG uses DS.

If *range* is omitted the first time the D command is used, the display starts at the target program's CS:IP registers. If *range* was specified in a preceding D command, the memory address following the last address displayed by that command is used. If a length is not explicitly stated in a D command, 128 bytes are displayed.

Each line displays a segment and offset, followed by the contents of 16 bytes of memory represented as hexadecimal values and separated by spaces (except the eighth and ninth values, which are separated by a dash), followed by the ASCII character equivalents (if any) of the same 16 bytes. In the ASCII portion, nonprinting characters are displayed as periods.

## Examples

To display the contents of the 128 bytes of memory beginning at 7F00:0100H, type

```
-D 7F00:0100  <Enter>
```

The contents of the memory addresses are displayed in the following format:

```
7F00:0100  20 64 65 76 69 63 65 0D-0A 00 60 39 0D 0A 00 7C    device...'9...|
7F00:0110  39 08 20 08 00 81 39 04-1B 5B 32 4A 42 BD 11 44   9. ...9..[2JB=.D
7F00:0120  2E 26 45 AF 11 47 B3 11-48 A5 11 4C B8 11 4E D3   .&E/.G3.H%.L8.NS
7F00:0130  11 50 DF 11 51 AB 11 54-DF 1E 56 37 11 5F 9F 16   .P_.Q+.T_.V7._..
7F00:0140  24 C0 11 00 03 4E 4F 54-C1 07 0A 45 52 52 4F 52   $@...NOTA..ERROR
7F00:0150  4C 45 56 45 4C 85 08 05-45 58 49 53 54 18 08 00   LEVEL...EXIST...
7F00:0160  03 44 49 52 03 91 0C 06-52 45 4E 41 4D 45 01 C0   .DIR....RENAME.@
7F00:0170  0F 03 52 45 4E 01 C0 0F-05 45 52 41 53 45 01 68   ..REN.@..ERASE.h
```

To view the next 128 bytes of memory, type

```
-D  <Enter>
```

In this case, the contents of memory addresses 7F00:0180H through 7F00:01FFH are displayed.

# DEBUG: E

Enter Data

## Purpose

Enters data into memory.

## Syntax

E *address* [*list*]

where:

| | |
|---|---|
| *address* | is the first memory location for data entry. |
| *list* | specifies the data to be entered into successive bytes of memory, starting at *address*. |

## Description

The Enter Data (E) command allows data to be entered into successive memory locations. The data can be entered in either hexadecimal or ASCII format. Data previously stored in the specified locations is lost.

The *address* parameter specifies the first byte to be modified. If *address* does not include a segment, DEBUG uses DS. The address is incremented for each byte of data stored.

The *list* parameter is one or more hexadecimal byte values and/or strings, separated by spaces, commas, or tab characters. Strings must be enclosed within single or double quotation marks, and case is significant within a string.

If *list* is included in the command line, the changes to memory are made unless an error is detected in the command line, in which case an error message is displayed and the E command is terminated. If *list* is omitted from the command line, the user is prompted byte by byte for data to be entered into memory, starting at *address*. The current contents of a byte are displayed, followed by a period. A new value for that byte can be entered as one or two hexadecimal digits (extra characters are ignored) or the contents can be left unchanged. Pressing the spacebar displays the contents of the next byte. Entering a minus sign or hyphen character (-) instead of pressing the spacebar displays the contents of the previous byte. A maximum of 8 bytes can be entered on each input line; a new line is begun each time an 8-byte boundary is crossed. Pressing the Enter key without pressing the spacebar or entering any data terminates data entry.

Text strings can be entered only by using the *list* parameter; they cannot be entered in response to an address prompt.

## Examples

To store the byte values 00H, 0DH, and 0AH in the three bytes beginning at DS:1FB3H, type

```
-E 1FB3 00 0D 0A  <Enter>
```

To store the string *MAIN MENU* into memory beginning at address ES:0C14H, type

```
-E ES:C14 "MAIN MENU"  <Enter>
```

# DEBUG: F

Fill Memory

## Purpose

Stores a repetitive data pattern in an area of memory.

## Syntax

F *range list*

where:

| | |
|---|---|
| *range* | is the starting and ending addresses or starting address and length of the memory to be filled. |
| *list* | is the data to be entered. |

## Description

The Fill Memory (F) command fills an area of memory with the data from a list. The data can be entered in either hexadecimal or ASCII format. Any data previously stored at the specified locations is lost. If an error message is displayed, the original values in memory remain unchanged.

The *range* parameter specifies the starting and ending addresses or the starting address and hexadecimal length in bytes of the area of memory to be filled. If *range* does not specify a segment, DEBUG uses DS.

The *list* parameter specifies one or more hexadecimal byte values and/or strings, separated by spaces, commas, or tab characters. Strings must be enclosed in single or double quotation marks, and case is significant within a string.

If the area to be filled is larger than the data list, the list is repeated as often as necessary to fill the area. If the data list is longer than the area of memory to be filled, it is truncated to fit into the area.

## Examples

To fill the area of memory from DS:0B10H through DS:0B4FH with the value 0E8H, type

```
-F B10 B4F E8  <Enter>
```

or

```
-F B10 L40 E8  <Enter>
```

To fill the 16 bytes of memory beginning at address CS:1FA0H by replicating the 2-byte sequence 0DH 0AH, type

```
-F CS:1FA0 1FAF 0D 0A  <Enter>
```

or

```
-F CS:1FA0 L10 0D 0A  <Enter>
```

To fill the area of memory from ES:0B00H through ES:0BFFH by replicating the text string *BUFFER*, type

```
-F ES:B00 BFF "BUFFER"  <Enter>
```

or

```
-F ES:B00 L100 "BUFFER"  <Enter>
```

# DEBUG: G

Go

## Purpose

Transfers control from DEBUG to the program being debugged.

## Syntax

G [=*address*] [*break0* [... *break9*]]

where:

| | |
|---|---|
| *address* | is the location DEBUG begins execution. |
| *break0*... *break9* | specify from 1 to 10 temporary breakpoints. |

## Description

The Go (G) command transfers control from DEBUG to the program being debugged. If no breakpoints are set, the program executes until it crashes or finishes, in which latter case the message *Program terminated normally* is displayed and control returns to DEBUG. (After this message is displayed, the program may need to be reloaded before it can be executed again.)

The *address* parameter can specify any location in memory. If no segment is specified, DEBUG uses the target program's CS register. If *address* is omitted, DEBUG transfers to the current address in the target program's CS:IP registers. An equal sign (=) must precede *address* to distinguish it from the breakpoints *break0*... *break9*.

The parameters *break0*... *break9* are addresses that represent from 1 to 10 temporary breakpoints that can be set as part of the G command. A breakpoint is an address at which execution stops. Breakpoints can be placed in any order, because execution stops at the first breakpoint address encountered, regardless of the position of that breakpoint in the list. Each breakpoint address must contain the first byte of an 8086 opcode. DEBUG installs breakpoints by replacing the first byte of the machine instruction at each breakpoint address with an INT 03H instruction (opcode 0CCH). If the program encounters a breakpoint, execution is suspended and control returns to DEBUG. DEBUG then restores the original machine code to the breakpoint addresses; displays the contents of the registers, the status of the flags, and the instruction pointed to by CS:IP; and displays the DEBUG prompt. If the program executes to completion without encountering any of the breakpoints or stops for any reason other than because it encountered a breakpoint, DEBUG does not replace the INT 03H instructions with the original machine code, and the Load File or Sectors (L) command must be used to reload the original program.

The G command requires that the target program's SS:SP registers point to a valid stack that has at least 6 bytes of stack space available. When the G command is executed, it

pushes the target program's flags and CS and IP registers onto the stack and then transfers control to the target program with an IRET instruction. Thus, if the target program's stack is not valid or is too small, the system may crash.

## Examples

To begin execution of the program in DEBUG's buffer at location CS:110AH and set breakpoints at CS:12FCH and CS:1303H, type

```
-G =110A 12FC 1303  <Enter>
```

To resume execution of the program after a breakpoint has been encountered and control has been returned to DEBUG, type

```
-G  <Enter>
```

## Messages

### bp Error

More than 10 breakpoints were specified in a G command. The command must be entered again with 10 or fewer breakpoints.

### Program terminated normally

No breakpoints were encountered and the target program executed to completion. If breakpoints were set, the original program should be restored with the L command.

# DEBUG: H

Perform Hexadecimal Arithmetic

## Purpose

Displays the sum and difference of two hexadecimal numbers.

## Syntax

H *value1 value2*

where:

*value1* and *value2* are any two hexadecimal numbers from 0 through FFFFH.

## Description

The Perform Hexadecimal Arithmetic (H) command displays the sum and the difference of two 16-bit hexadecimal numbers — that is, the result of the operations *value1+value2* and *value1–value2*. If *value2* is greater than *value1*, the difference of the two values is displayed as a two's complement number. This command is convenient for quickly calculating addresses and other values during an interactive debugging session.

## Examples

To display the sum and the difference of the values 4B03H and 104H, type

```
-H 4B03 104  <Enter>
```

This produces the following display:

```
4C07  49FF
```

If the addition produces an overflow, the four least significant digits are displayed. For example, the command line

```
-H FFFF 2  <Enter>
```

produces the following display:

```
0001 FFFD
```

If the second number is bigger than the first, the difference is displayed in two's complement form. For example, the command line

```
-H 1 2  <Enter>
```

produces the following display:

```
0003 FFFF
```

# DEBUG: I

Input from Port

## Purpose

Reads and displays 1 byte from an input/output (I/O) port.

## Syntax

I *port*

where:

*port* is an I/O port address from 0 through FFFFH.

## Description

The Input from Port (I) command reads the specified I/O port address and displays the data as a two-digit hexadecimal number.

***Warning:*** The I command should be used with caution because it directly accesses the computer hardware and no error checking is performed. Input operations directed to the ports assigned to some peripheral device controllers may interfere with the proper operation of the system. If no device has been assigned to the specified I/O port or if the port is write-only, the value displayed by an I command is unreliable.

## Example

To read and display the contents of I/O port 10AH, type

```
-I 10A  <Enter>
```

An example of the output of this command is

```
FF
```

# DEBUG: L

Load File or Sectors

## Purpose

Loads a file or individual sectors from a disk into DEBUG's memory.

## Syntax

L [*address*]

or

L *address drive start number*

where:

| | |
|---|---|
| *address* | is the memory location for the data to be read from the disk. |
| *drive* | is the number of the disk drive to read (0 = drive A, 1 = drive B, 2 = drive C, and so on). |
| *start* | is the hexadecimal number of the first logical sector to load (0–FFFFH). |
| *number* | is the hexadecimal number of consecutive sectors to load (0–FFFFH). |

## Description

The Load File or Sectors (L) command loads a file or individual sectors from a disk. When the L command is entered without parameters or with only an address, the file specified in the DEBUG command line or the one in the most recent Name File or Command-Tail Parameters (N) command line is loaded from the disk into memory. If no segment is specified in *address*, DEBUG uses CS. If the file's extension is .EXE, the file is placed in DEBUG's target program buffer at the load address specified in the .EXE file's header. If the file's extension is .COM, the file is loaded at offset 100H. (If for some reason an address other than 100H is entered for a .EXE or .COM file, an error message is displayed; if the address is 100H, the specification is ignored.) The length of the file or, in the case of a .EXE file, the actual length of the program (the length of the file minus the header) is placed in the target program's BX and CX registers, with the most significant 16 bits in register BX.

The L command can also be used to bypass the MS-DOS file system and directly access logical sectors on the disk. The memory address (*address*), disk drive number (*drive*), starting logical sector number (*start*), and number of sectors to load (*number*) must all be specified in the command line.

***Note:*** The L command should not be used to access logical sectors on network drives.

## Examples

To load the file specified in the DEBUG command line or in the most recent N command into DEBUG's target program buffer, type

```
-L  <Enter>
```

To load eight sectors from drive B, starting at logical sector 0, to memory location CS:0100H, type

```
-L 100 1 0 8  <Enter>
```

## Messages

**Disk error reading drive *X***
The specified drive does not exist or the disk in the specified drive is defective.

**File not found**
The file specified in the most recent N command cannot be found.

# DEBUG: M

Move (Copy) Data

## Purpose

Copies the contents of one area of memory to another.

## Syntax

M *range address*

where:

| | |
|---|---|
| *range* | specifies the starting and ending addresses or the starting address and length of the area of memory to be copied. |
| *address* | is the first byte in which the copied data will be placed. |

## Description

The Move (Copy) Data (M) command copies data from one memory location to another without altering the data in the original location. If the source and destination areas overlap, the data is copied so that the resulting copy is correct; the data in the *original* location is changed where the two areas overlap.

The *range* parameter specifies either the starting and ending addresses or the starting address and length of the memory to be copied. The *address* parameter is the first byte in which the copy will be placed. If *range* does not contain an explicit segment, DEBUG uses DS; if *address* does not contain a segment, DEBUG uses the segment used for *range*.

## Example

To copy the data in locations DS:0800H through DS:08FFH to locations DS:0900H through DS:09FFH, type

```
-M 800 8FF 900  <Enter>
```

or

```
-M 800 L100 900  <Enter>
```

# DEBUG: N

Name File or Command-Tail Parameters

## Purpose

Inserts filenames and/or switches into the simulated program segment prefix (PSP).

## Syntax

N *parameter* [*parameter*...]

where:

*parameter* is one or more filenames or switches to be placed in the simulated PSP.

## Description

The Name File or Command-Tail Parameters (N) command is used to enter one or more parameters into the simulated PSP that is built at the base of the buffer holding the program to be debugged. The N command can also be used before the Load File or Sectors (L) and Write File or Sectors (W) commands to name the file to be read from or written to a disk.

The count of the characters following the N command is placed at DS:0080H in the simulated PSP, and the characters themselves are copied into the PSP starting at offset 81H. The string is terminated by a carriage return (0DH), which is not included in the count. If the first and second parameters follow the naming conventions for MS-DOS files, they are parsed into the default file control blocks (FCBs) in the simulated PSP at offsets 5CH and 6CH, respectively. (Switches specified as parameters are stored in the PSP starting at offset 81H along with the rest of the command line but are not included in the FCBs.)

If the N command line contains only one filename, any parameters placed in the default FCBs by a previous N command are destroyed. If the drive specified with the first filename parameter is invalid, the AL register is set to 0FFH. If the drive specified with the second filename parameter is invalid, the AH register is set to 0FFH. The existence of a file specified with the N command is not verified until it is loaded with the L command.

## Examples

Assume that DEBUG was started without specifying the name of a target program in the command line. To load the program CLEAN.COM for execution under the control of DEBUG, use the N and L commands together as follows:

```
-N CLEAN.COM  <Enter>
-L  <Enter>
```

Then, to place the parameter MYFILE.DAT in the simulated PSP's command tail and parse MYFILE.DAT into the first default FCB, type

```
-N MYFILE.DAT  <Enter>
```