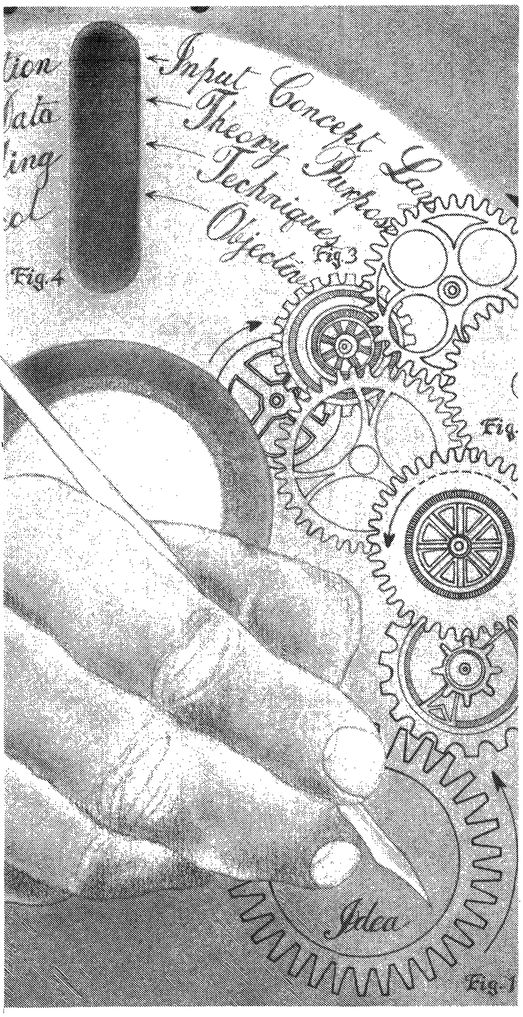

The Waite Group's MS-DOS® Developer's Guide

Quick Reference Card

Owned or Sponsored by:

Predefined File Handles

0	= Standard input device (can be redirected)
1	= Standard output device (can be redirected)
2	= Standard error device (cannot be redirected)
3	= Standard auxiliary device
4	= Standard printer device

Error Returns

Of the following error codes, only codes 01h through 12h are returned in AX upon exit from int 21h or 24h. The rest are obtained by issuing the Get Extended Error function call (int 21h, function 59h).

MS-DOS 2.0 through 4.0 Error Codes

01h	= Invalid function number
02h	= File not found
03h	= Path not found
04h	= Too many open files (no handles left)
05h	= Access denied
06h	= Invalid handle
07h	= Memory control blocks destroyed
08h	= Insufficient memory
09h	= Invalid memory block address
0Ah	= Invalid environment
0Bh	= Invalid format
0Ch	= Invalid access code
0Dh	= Invalid data
0Eh	= Reserved
0Fh	= Invalid drive was specified
10h	= Attempt to remove the current directory
11h	= Not same device
12h	= No more files
13h	= Attempt to write on a write-protected diskette
14h	= Unknown unit
15h	= Drive not ready
16h	= Unknown command
17h	= CRC error
18h	= Bad request structure length
19h	= Seek error
1Ah	= Unknown media type
1Bh	= Sector not found
1Ch	= Printer out of paper
1Dh	= Write fault
1Eh	= Read fault
1Fh	= General failure

MS-DOS 3.0 through 4.0 Error Codes

20h	= Sharing violation
21h	= Lock violation
22h	= Invalid disk change
23h	= FCB unavailable
24h	= Sharing buffer overflow (MS-DOS 3.3, 4.0)
25h–41h	= (Reserved)
42h–58h	= (See MS-DOS 3.1 through 4.0 Error Codes)
59h–5Fh	= (Reserved)
60h	= File exists
61h	= (Reserved)
62h	= Cannot make function
63h	= Failure on int 24h
64h–68h	= (See MS-DOS 3.3, 4.0 Error Codes)

MS-DOS 3.1 through 4.0 Error Codes

42h	= Network request not supported
43h	= Remote computer not listening
44h	= Duplicate name on network
45h	= Network name not found
46h	= Network busy

47h	= Network device no longer exists
48h	= NETBIOS command limit exceeded
49h	= Network adapter hardware error
4Ah	= Incorrect response from network
4Bh	= Unexpected network error
4Ch	= Incompatible remote adapter
4Dh	= Print queue full
4Eh	= Queue not full
4Fh	= Not enough space to print file
50h	= Network name was deleted
51h	= Access denied
52h	= Network device type incorrect
53h	= Network name not found
54h	= Network name limit exceeded
55h	= NETBIOS session limit exceeded
56h	= Temporarily paused
57h	= Network request not accepted
58h	= Print/disk redirection paused
59h–5Fh	= (Reserved)
60h	= File exists
61h	= Reserved
62h	= Cannot make
63h	= Fail on int 24

MS-DOS 3.3, 4.0 Error Codes

64h	= Out of structures
65h	= Already assigned
66h	= Invalid password
67h	= Invalid parameter
68h	= Network write fault

Error Classes	
01h = Out of resource	08h = Not found
02h = Temporary situation	09h = Bad format
03h = Authorization	0Ah = Locked
04h = Internal	0Bh = Media failure
05h = Hardware failure	0Ch = Already exists
06h = System failure	0Dh = Unknown
07h = Application error	

Action Codes	
01h = Retry	05h = Immediate exit
02h = Delay retry	06h = Ignore
03h = Reenter input	07h = User intervention
04h = Abort	

Locus	
01h = Unknown	04h = Serial device
02h = Block device	05h = Memory
03h = Reserved	

MS-DOS Interrupts

NOTE: In the following descriptions of MS-DOS interrupts, the numbers in brackets refer to versions of MS-DOS.

Interrupt 20h—Program Terminate [1][2][3][4]

ENTRY: CS = Segment address of program's PSP
RETURN: None

Interrupt 21h—Function Call Request

NOTE: Unless otherwise noted, all functions check for Ctrl-Break and Ctrl-C; if issued, interrupt 23h is executed.

AH = 00h Program Terminate [1][2][3][4]

ENTRY: CS = Segment address of program's PSP
RETURN: None
NOTE: All file buffers are flushed: files opened with FCBs may have data lost if not closed beforehand. Func. 4Ch is preferred.

AH = 01h Input Character from Console with Echo [1][2][3][4]

ENTRY: None
RETURN: If AL 〉 0 on first call, AL = standard ASCII character
If AL = 0 on first call, call function 01h second time to obtain Extended ASCII character in AL

AH = 02h Output Character to Console [1][2][3][4]

ENTRY: DL = character to write to first serial port [1] or to STDAUX [2][3][4]
RETURN: None

AH = 03h Input Character from Auxiliary Port [1][2][3][4]

ENTRY: None
RETURN: AL = Character from first serial port [1] or from STDAUX [2][3][4]
NOTE: Input is not buffered or interrupt-driven. The status of the serial port is not checked (see ROM-BIOS int 14h).

AH = 04h Output Character to Auxiliary Port [1][2][3][4]

ENTRY: DL = Character to output to STDAUX
RETURN: None
NOTE: The status of the serial port is not checked.

AH = 05h Output Character to Printer [1][2][3][4]

ENTRY: DL = Character to output to STDPRN
RETURN: None

AH = 06h Direct Console I/O [1][2][3][4]

ENTRY: If DL 〈 〉 0FFh, output character in DL to STDOUT; otherwise perform direct console input
RETURN: None for direct console output.
For direct console input:
ZF = 1 if no character available; else AL = character
NOTE: Extended ASCII codes require two calls.

AH = 07h Direct Input Character from Console without Echo [1][2][3][4]

ENTRY: None
RETURN: AL = Character from STDIN
NOTE: Functions 07h and 08h require 2 calls for Extended ASCII codes

AH = 08h Input Character from Console without Echo [1][2][3][4]

ENTRY: None
RETURN: AL = Character from STDIN

AH = 09h Output String to Console [1][2][3][4]

ENTRY: DS:DX = Pointer to string terminated by "$"
RETURN: None

AH = 0Ah Input Buffered String from Console with Echo [1][2][3][4]

ENTRY: DS:DX = Pointer to input buffer. Buffer structure:

```
buf_count  db ?              ; number of bytes in buffer
ret_count  db ?              ; number of bytes returned
ret_char_str db x DUP (?)    ; returned characters
```

RETURN: None

AH = 0Bh Check Standard Input Status [1][2][3][4]

ENTRY: None

RETURN: AL = 0FFh if character available from STDIN; AL ⟨⟩ FFh if not

AH = 0Ch Clear Keyboard Buffer and Invoke Keyboard Function [1][2][3][4]

ENTRY: AL = int 21h function number (01h, 06h, 07h, 08h or 0Ah)
Other registers defined by function in AL

RETURN: AL = Character (unless function 0Ah was invoked)
Other registers defined by function in AL on entry

AH = 0Dh Disk Reset [1][2][3][4]

ENTRY: None

RETURN: None

NOTE: Flushes all file buffers but doesn't close files.

AH = 0Eh Select Disk [1][2][3][4]

ENTRY: DL = Drive number (0 = A:, . . . , 26 = Z:)

RETURN: AL = Number of logical drives (0 = A:, . . . , 26 = Z:)

NOTE: In DOS 3 and 4, a minimum of 5 logical drives is reported unless overridden by LASTDRIVE setting in CONFIG.SYS.

AH = 0Fh FCB Open File [1][2][3][4]

ENTRY: DS:DX = Pointer to unopened FCB

RETURN: AL = 00h if file was opened successfully; AL = 0FFh if not

AH = 10h FCB Close File [1][2][3][4]

ENTRY: DS:DX = Pointer to opened FCB

RETURN: AL = 00h if file was closed successfully; AL = 0FFh if not

AH = 11h FCB Search for First Entry [1][2][3][4]

ENTRY: DS:DX = Pointer to an unopened FCB

RETURN: AL = 00h if match was found; AL = 0FFh if not

AH = 12h FCB Search for Next Entry [1][2][3][4]

ENTRY: DS:DX = Pointer to FCB returned by previous search-first or search-next function call

RETURN: AL = 00h if match was found; AL = 0FFh if not

AH = 13h FCB Delete File [1][2][3][4]

ENTRY: DS:DX = Pointer to an unopened FCB

RETURN: AL = 00h if file was deleted; AL = 0FFh if not

AH = 14h FCB Sequential Read [1][2][3][4]

ENTRY: DS:DX = Pointer to an opened FCB

RETURN: AL = Success/failure
00h = read was successfully completed
01h = no read attempted; already at end of file
02h = read cancelled; DTA too small
03h = partial read completed; now at EOF

AH = 15h FCB Sequential Write [1][2][3][4]

ENTRY: DS:DX = Pointer to an opened FCB

RETURN: AL = Success/failure
00h = write was successfully completed
01h = no write attempted; media is full
02h = write cancelled; DTA too small

AH = 16h FCB Create File [1][2][3][4]

ENTRY: DS:DX = Pointer to an unopened FCB

RETURN: AL = 00h if file was created; AL = 0FFh if not

AH = 17h FCB Rename File [1][2][3][4]

ENTRY: DS:DX = Pointer to a modified FCB (new name starts in current block number field)

RETURN: AL = 00h if file was renamed; AL = 0FFh if not

AH = 19h Get Current Disk [1][2][3][4]

ENTRY: None

RETURN: AL = Current drive number (0 = A:, . . . , 25 = Z:)

AH = 1Ah Set Disk Transfer Address [1][2][3][4]

ENTRY: DS:DX = Pointer to new DTA

RETURN: None

AH = 1Bh Get Allocation Table Information [1][2][3][4]

ENTRY: None

RETURN: DS:BX = Pointer to byte containing FAT ID byte for default drive
DX = Number of clusters
AL = Number of sectors per cluster
CX = Number of bytes per sector

AH = 1Ch Get Allocation Table Information for Specific Device [1][2][3][4]

ENTRY: DL = Drive number (0 = current drive, 1 = A:, . . . , 26 = Z:)

RETURN: Same as for Function 1Bh

AH = 21h Random Read [1][2][3][4]

ENTRY: DS:DX = Pointer to an opened FCB

RETURN: AL = Return status:
00h = read was successful
01h = end of file; no data read
02h = DTA is too small
03h = end of file; partial record read

AH = 22h Random Write [1][2][3][4]

ENTRY: DS:DX = Pointer to an opened FCB

RETURN: AL = Return status:
00h = write was successful
01h = no write attempted; media full
02h = write cancelled; DTA too small

AH = 23h Get File Size [1][2][3][4]

ENTRY: DS:DX = Pointer to an unopened FCB

RETURN: If AL = 00h, FCB random record field = records in file
If AL = 0FFh, file not found

AH = 24h Set Relative Record Field [1][2][3][4]

ENTRY: DS:DX = Pointer to an opened FCB

RETURN: None

AH = 25h Set Interrupt Vector [1][2][3][4]

ENTRY: AL = Interrupt number to set
DS:DX = Pointer to new interrupt handling routine

RETURN: None

AH = 26h Create New Program Segment Prefix [1][2][3][4]

ENTRY: DX:0 = Pointer to new PSP area

RETURN: None

AH = 27h Random Block Read [1][2][3][4]

ENTRY: DS:DX = Pointer to an opened FCB
CX = Number of records to read

RETURN: AL = Return status:
00 = read was successful
01 = end of file; no data read
02 = DTA too small
03 = end of file; partial record read
CX = Actual number of records read

AH = 28h Random Block Write [1][2][3][4]

ENTRY: DS:DX = Pointer to an opened FCB
CX = Number of records to be written

RETURN: AL = Return status:
00h = write was successful
01h = no write attempted; media full
02h = write cancelled; DTA too small
CX = Actual number of records written

AH = 29h FCB Parse Filename [1][2][3][4]

ENTRY: DS:SI = Pointer to a command line to parse
ES:DI = Pointer to FCB for parsed filename
AL = Parsing control

```
76543210
││││││││
│││││││└── 1 = scan off leading separators
││││││└─── 1 = set drive ID byte*
│││││└──── 1 = set filename*
││││└───── 1 = set extension*
└┴┴┴────── Reserved
```

*only if specified on the command line

RETURN: DS:SI = Pointer to first byte after parsed filename
ES:DI = Pointer to first byte of the formatted FCB
AL = Return status:
00h = no global characters encountered
01h = global characters were encountered
0FFh = drive specified was invalid

AH = 2Ah Get Date [1][2][3][4]
ENTRY: None
RETURN: CX = Year (1980 to 2099)
DH = Month (1 to 12)
DL = Day (1 to 31)
AL = Day of the week (0 = Sunday)

AH = 2Bh Set Date [1][2][3][4]
ENTRY: CX = Year (1980 to 2099)
DH = Month (1 to 12)
DL = Day (1 to 31)
RETURN: AL = 00h if date was valid; AL = 0FFh if not valid

AH = 2Ch Get Time [1][2][3][4]
ENTRY: None
RETURN: CH = Hour (0 to 23)
CL = Minutes (0 to 59)
DH = Seconds (0 to 59)
DL = Hundredths (0 to 99)

AH = 2Dh Set Time [1][2][3][4]
ENTRY: CH = Hour (0 to 23)
CL = Minutes (0 to 59)
DH = Seconds (0 to 59)
DL = Hundredths (0 to 99)
RETURN: AL = 00h if time was valid; AL = 0FFh if not valid

AH = 2Eh Set/Reset Verify Switch [1][2][3][4]
ENTRY: AL = 00h to set verify to off; AL = 01h to set verify to on
RETURN: None

AH = 2Fh Get Disk Transfer Address (DTA) [2][3][4]
ENTRY: None
RETURN: ES:BX = Pointer to the current DTA

AH = 30h Get MS-DOS Version Number [2][3][4]
ENTRY: None
RETURN: AL = Major version number (left of decimal)
AH = Minor version number (right of decimal)
BX, CX = 0000
NOTE: AX = 0 if MS-DOS version 1.X

AH = 31h Terminate Process and Remain Resident [2][3][4]
ENTRY: AL = Return code (batch ERRORLEVEL)
DX = Number of memory paragraphs to stay resident
RETURN: None

AH = 33h Get/Set Ctrl-Break Check State [2][3][4]
ENTRY: AL = Get current state; AL = Set Ctrl-Break check
DL = 00h to set Ctrl-Break to off; AL = 01h to set to on
RETURN: DL = 00h if Ctrl-Break is off; AL = 01h if on

AH = 35h Get Interrupt Vector [2][3][4]
ENTRY: AL = Vector number
RETURN: ES:BX = Pointer to the current interrupt handler

AH = 36h Get Disk Free Space [2][3][4]
ENTRY: DL = Drive number (0 = current drive, 1 = A:, . . . , 26 = Z:)
RETURN: BX = Number of available clusters
DX = Number of clusters on drive
CX = Number of bytes per sector
If AX = 0FFFFh, drive is invalid
If AX ⟨ ⟩ 0FFFFh, AX = number of sectors per cluster

AH = 38h Get Current Country Information [2][3][4]
ENTRY: AL = 00 to get current country information
AL = 01h through 0FEh for country codes ⟨255
AL = 0FFh for country codes ⟩255
BX = Country code if AL = 0FFh
DS:DX = Pointer to 34-byte country information buffer
RETURN: If CF = 0, BX = country code
If CF = 1, AX = error code
NOTE: See MS-DOS manual for structure and contents of country information buffer.

AH = 38h Set Country Information [3][4]
ENTRY: DX = 0FFFFh (to indicate "set country")
AL = 01h through 0FEh for country codes ⟨255
AL = 0FFh for country codes ⟩255
BX = Country code if AL = 0FFh
RETURN: If CF = 1, AX = Error code

AH = 39h Create Subdirectory (MKDIR) [2][3][4]
ENTRY: DS:DX = Pointer to ASCIIZ path name
RETURN: If CF = 1, AX = error

AH = 3Ah Remove Subdirectory (RMDIR) [2][3][4]
ENTRY: DS:DX = Pointer to ASCIIZ path name
RETURN: If CF = 1, AX = error

AH = 3Bh Change Current Directory (CHDIR) [2][3][4]
ENTRY: DS:DX = Pointer to ASCIIZ path name
RETURN: If CF = 1, AX = error

AH = 3Ch Create a File (CREAT) [2][3][4]
ENTRY: DS:DX = Pointer to ASCIIZ path name
CX = File attributes
RETURN: If CF = 0, AX = file's handle
If CF = 1, AX = error code

AH = 3Dh Open a File [2][3][4]
ENTRY: DS:DX = Pointer to an ASCIIZ path name
AL = Open mode:

```
76543210
││││││││
│││││└┴┴── Access mode: 000 = read only
│││││                    001 = write only,
│││││                    010 => read/write
││││└───── Always 0
│└┴┴────── Sharing mode:
│             000 = compatibility mode
│             001 = deny read/write
│             010 = deny write
│             011 = deny read
│             100 = deny none
└───────── Inheritance flag
```

RETURN: If CF = 0, AX = file handle
If CF = 1, AX = error code

NOTE: Opening of network files not available under DOS 2.X.

AH = 3Eh Close a File Handle [2][3][4]
ENTRY: BX = File handle
RETURN: If CF = 1, AX = error code

AH = 3Fh Read from a File or Device [2][3][4]
ENTRY: BX = File handle
CX = Number of bytes to read
DS:DX = Pointer to read buffer
RETURN: If CF = 0, AX = number of bytes actually read
If CF = 1, AX = error code

AH = 40h Write to a File or Device [2][3][4]
ENTRY: BX = File handle
CX = Number of bytes to write
DS:DX = Pointer to write buffer
RETURN: If CF = 0, AX = number of bytes actually written
If CF = 1, AX = error code

AH = 41h Delete a File from a Specified Directory (UNLINK) [2][3][4]
ENTRY: DS:DX = Pointer to an ASCIIZ filename
RETURN: If CF = 1, AX = error code

AH = 42h Move File Read/Write Pointer (LSEEK) [2][3][4]
ENTRY: CX:DX = Distance to move in bytes (offset)
AL = Origin of move:
00 = beginning of file plus offset
01 = current location plus offset
02 = end of file plus offset
BX = File's handle
RETURN: If CF = 0, DX:AX = new pointer location
If CF = 1, AX = error code

AH = 43h Change File Mode (CHMOD) [2][3][4]
ENTRY: DS:DX = Pointer to an ASCIIZ path name
AL = 00h to get attribute; AL = 01h to set attribute
CH = 00h if AL = 01h
CL = New attribute if AL = 01h
RETURN: If CF = 0 and AL = 00h, CL = file's attributes
If CF = 1, AX = error code

AH = 44h I/O Control for Devices (IOCTL)
NOTE: See the MS-DOS technical reference manual for details on the following IOCTL subfunctions:

00h	Get device information [2][3][4]
01h	Set device information [2][3][4]
02h	Read from character device [2][3][4]
03h	Write to character device [2][3][4]
04h	Read from block device [2][3][4]
05h	Write to block device [2][3][4]
06h	Get input status [2][3][4]
07h	Get output status [2][3][4]
08h	Is a particular block device changeable [3][4]
09h	Is logical device local or remote [3.1][3.2][3.3][4]
0Ah	Is handle local or remote [3.1][3.2][3.3][4]
0Bh	Change sharing retry count [3][4]
0Ch	Generic IOCTL handle request (code page switching) [3.3][4]
0Dh	Block device generic IOCTL request [3.2][3.3][4]
0Eh	Get logical device [3.2][3.3][4]
0Fh	Set logical device [3.2][3.3][4]

AH = 45h Duplicate a File Handle (DUP) [2][3][4]
ENTRY: BX = Existing file handle
RETURN: If CF = 0, AX = new duplicate file handle
If CF = 1, AX = error code

AH = 46h Force a Duplicate of a File Handle (FORCDUP) [2][3][4]
ENTRY: BX = Existing file handle
CX = Desired duplicate file handle
RETURN: If CF = 1, AX = Error code

AH = 47h Get Current Directory [2][3][4]
ENTRY: DS:SI = Pointer to a 64-byte user buffer
DL = Drive number (0 = current drive, 1 = A:, . . . , 26 = Z:)
RETURN: DS:SI = Pointer to full path name from root
If CF = 1, AX = Error code
NOTE: Returned path name does not include drive ID and leading "\".

AH = 48h Allocate Memory [2][3][4]
ENTRY: BX = Number of paragraphs of memory requested
RETURN: If CF = 0, AX:0 = pointer to allocated memory block
If CF = 1, AX = error code and BX = size of the largest block of memory available (paragraphs)

AH = 49h Free Allocated Memory [2][3][4]
ENTRY: ES = Segment of allocated block to be freed
RETURN: IF CF = 1, AX = error code

AH = 4Ah Modify Allocated Memory Blocks (SETBLOCK) [2][3][4]
ENTRY: ES:0 = Segment address of allocated block to be modified
BX = New number of paragraphs for block
RETURN: If CF = 1, AX = error code and BX = maximum size possible for block

AH = 4Bh Load or Execute a Program (EXEC) [2][3][4]
ENTRY: DS:DX = Pointer to an ASCIIZ file specification
AL = Function value:
00h = load and execute the program
03h = load an overlay
ES:BX = Pointer to parameter block:
If AL = 00h

```
seg_env  dw  ?  ; segment of envir. string
cmd_ptr  dd  ?  ; pointer to command line
fcb1_ptr dd  ?  ; pointer to first FCB
fcb2_ptr dd  ?  ; pointer to second FCB
```

If AL = 03h

```
seg_load dw  ?  ; segment at which to load file
rel_fact dw  ?  ; relocation factor to be used
```

RETURN: If CF = 1, AX = error code

AH = 4Ch Terminate a Process (EXIT) [2][3][4]
ENTRY: AL = Return code (batch ERRORLEVEL)
RETURN: None

AH = 4Dh Get Return Code of a Subprocess (WAIT) [2][3][4]
ENTRY: None
RETURN: AL = Return code sent by subprocess
AH = Return status:
00h = normal termination
01h = Ctrl-Break termination
02h = critical error termination
03h = stayed resident via int 21h function 31h

AH = 4Eh Find First Matching File (FINDFIRST) [2][3][4]

ENTRY: DS:DX = Pointer to ASCIIZ file specification
CX = Attribute used during search

RETURN: If CF = 1, AX = Error code
If CF = 0, DTA is filled as follows:

```
reserved db 21 dup (?) ; reserved
attrib   db ?          ; file's attribute
time     dw ?          ; file's time stamp
date     dw ?          ; file's date stamp
size     dd ?          ; file's size
name     db 13 dup (?) ; ASCIIZ file name
```

AH = 4Fh Find Next Matching File (FINDNEXT) [2][3][4]

ENTRY: DTA as returned from previous FINDFIRST or FINDNEXT call

RETURN: Same as FINDFIRST function call

AH = 54h Get Verify Setting [2][3][4]

ENTRY: None

RETURN: AL = 00h if verify is off; AL = 01h if verify is on

AH = 56h Rename a File [2][3][4]

ENTRY: DS:DX = Pointer to old ASCIIZ [drive:path\filename]
ES:DI = Pointer to new ASCIIZ [drive:path\filename]

RETURN: If CF = 1, AX = error code

AX = 5700h Get a File's Date and Time [2][3][4]

ENTRY: BX = File's handle

RETURN: If CF = 0, CX = file's time and DX = file's date
If CF = 1, AX = error code

AX = 5701h Set a File's Date and Time [2][3][4]

ENTRY: BX = File's handle
CX = New time
DX = New date

RETURN: If CF = 1, AX = error code

AH = 59h Get Extended Error Information [3][4]

ENTRY: BX = 0000h

RETURN: AX = Extended error code
BH = Error class
BL = Suggested action
CH = Locus
CL, DX, SI, DI, ES and DS are destroyed.

AH = 5Ah Create a Temporary File [3][4]

ENTRY: DS:DX = Pointer to ASCIIZ string with drive and path, ending in "\"
CX = File attributes

RETURN: If CF = 0, AX = file handle and DS:DX = pointer to ASCIIZ string, complete with filename
If CF = 1, AX = error code

AH = 5Bh Create a New File [3][4]

ENTRY: DS:DX = Pointer to ASCIIZ path/filename
CX = File attributes

RETURN: If CF = 0, AX = handle
If CF = 1, AX = error code

AH = 5Ch Lock/Unlock File Access [3][4]

ENTRY: AL = to lock file access; AL = 01h to unlock file access
BX = File handle
CX = High word of offset
DX = Low word of offset
SI = High word of length
DI = Low word of length

RETURN: If CF = 1, AX = error code

AX = 5E00h NETWORK: Get Machine Name [3.1][3.2][3.3][4]

ENTRY: DS:DX = Pointer to 16-byte buffer for ASCIIZ computer name

RETURN: If CF = 0, DS:DX = pointer to ASCIIZ computer name
If CF = 1, AX = error code
If CH = 0, name/number is undefined
If CH ⟨ ⟩ 0, name/number is defined and CL = NETBIOS name number

AX = 5E02h NETWORK: Set Printer Setup String [3.1][3.2][3.3][4]

ENTRY: BX = Redirection list index
CX = Length of setup string (maximum length = 64 bytes)
DS:SI = Pointer to printer setup string

RETURN: If CF = 1, AX = error code

AX = 5E03h NETWORK: Get Printer Setup String [3.1][3.2][3.3][4]

ENTRY: BX = Redirection list index
ES:DI = Pointer to 64-byte printer setup buffer

RETURN: If CF = 0, CX = length of returned data and ES:DI = pointer to printer setup string
If CF = 1, AX = error code

AX = 5F02h NETWORK: Get Redirection List Entry [3.1][3.2][3.3][4]

ENTRY: BX = Redirection list index (zero-based)
DS:SI = Pointer to 128-byte buffer for local name
ES:DI = Pointer to 128-byte buffer for network name

RETURN: If CF = 0, BH = device status flag
  If bit 0 = 0, device is valid
  If bit 0 = 1, device is invalid
BL = Device type
CX = Stored parameter value
DS:SI = ASCIIZ local name
ES:DI = ASCIIZ network name
If CF = 1, AX = error code

AX = 5F03h NETWORK: Redirect Device [3.1][3.2][3.3][4]

ENTRY: BL = Device type:
03 = Printer device
04 = File device
CX = 0000h
DS:SI = Pointer to ASCIIZ local name to redirect
ES:DI = Pointer to ASCIIZ network destination name

RETURN: If CF = 1, AX = error code

AH = 62h Get Program Segment Prefix Address [3][4]

ENTRY: None

RETURN: BX:0 = Pointer to current PSP

AH = 65h Get Extended Country Information [3.3][4]

ENTRY: AL = Information ID
BX = Code page (−1 = global code page)
DX = Country ID (−1 = current country)
CX = Size
ES:DI = Pointer to country information buffer

RETURN: If CF = 0, CX = size of country information returned and ES:DI = pointer to country information
If CF = 1, AX = error code

AH = 66h Get/Set Global Code Page [3.3][4]
ENTRY: AL = 01h to get global code page; AL = 02h to set
BX = Code page (if AL = 02h)
RETURN: If CF = 0, BX = active code page and DX = system code page
If CF = 1, AX = error code
AH = 67h Set Handle Count [3.3][4]
ENTRY: BX = Number of open handles allowed
RETURN: If CF = 1, AX = error code
AH = 68h Commit File [3.3][4]
ENTRY: BX = File handle
RETURN: CF = 1, AX = error code
AH = 69h Extended Open/Create [4]
ENTRY: BX = Open mode:

```
BL=76543210
   ││││││││
   │││││└┴┴── Access code:
   │││││         000 = read
   │││││         001 = write
   │││││         002 = read/write
   ││└┴┴───── Sharing mode:
   ││            000 = compatibility
   ││            001 = deny read/write
   ││            010 = deny write
   ││            011 = deny read
   ││            100 = deny none
   │└──────── Inheritance:
   │             0 = pass handle to child
   │             1 = no inheritance
   └───────── Reserved

BH=76543210
   ││││││││
   ││└┴┴┴┴┴── Reserved
   │└──────── 0 = execute INT 24h
   │          1 = return error
   │───────── 0 = no commit
   │          1 = auto commit
   └───────── Reserved
```

CX = New file attributes (ignored on file open)
DX = Function control:

```
76543210
││││││││
││││└┴┴┴── Exists action:
││││          000 = fail
││││          001 = open
││││          010 = replace/open
└┴┴┴────── Does not exist action
              000 = fail
              001 = create
```

DS:SI = Pointer to 64-byte ASCIIZ file specification
RETURN: If CF = 0, AX = file handle and CX = action-taken code:
1 = file opened
2 = file created/opened
3 = file replaced/opened
If CF = 1, AX = error code

Interrupt 22h—Terminate Address [1][2][3][4]
NOTE: Don't issue this interrupt directly; instead, use the EXEC function call, which issues int 22h for you.

Interrupt 23h—Ctrl/Break Exit Address [1][2][3][4]
NOTE: Don't issue this interrupt directly; if BREAK is on, int 23h is checked on most function calls (except functions 06h and 07h).

Interrupt 24h—Critical Error Handler Address [3][4]
ENTRY: AH =

```
Bits 76543210
     ││││││││
     │││││││└── 0 read / 1 write operation
     │││││└┴─── affected disk area:
     │││││         00 = DOS area; 01 FAT area;
     │││││         10 = directory; 11 = data area
     ││││└───── FAIL allowed: 0 = no; 1 = yes
     │││└────── RETRY allowed: 0 = no; 1 = yes
     ││└─────── IGNORE allowed: 0 = no; 1 = yes
     │└──────── (unused)
     └───────── 0 = disk error; 1 = other
```

BP:SI = Pointer to device header control block from which additional information can be retrieved.
DL = Device error code, as follows:
RETURN: AL = 0 (ignore the error)
AL = 1 (retry the error)
AL = 2 (terminate the program through int 23h)
AL = 3 (system failure: call in progress)
Interrupt 24h Critical Error Handler Address Error Codes:
00h = Attempt to write on write-protected disk
01h = Unknown unit
02h = Drive not ready
03h = Unknown command
04h = Data error (CRC)
05h = Bad request structure length
06h = Seek error
07h = Unknown media type
08h = Sector not found
09h = Printer out of paper
0Ah = Write fault
0Bh = Read fault
0Ch = General failure

Interrupts 25h (Absolute =⟨32-Mbyte Disk Read) and 26h (Absolute =⟨32-Mbyte Disk Write) [1][2][3][4]
ENTRY: AL = Drive number (0 = A, 1 = B, etc.)
CX = Number of sectors to read (int 25h) or write (int 26h)
DX = Beginning logical sector number
DS:BX = Transfer address
RETURN: CF = 0 if successful transfer
CF = 1 if unsuccessful transfer:
AL = Error code
AH = 80h if attachment failed to respond
40h if SEEK operation failed
08h if bad CRC on disk read
04h if requested sector not found
03h if write attempt on write-protected diskette
02h if error other than types listed above
AX = 0207h if failed to read/write extended format using conventional int 25h/26h calls

Interrupts 25h (Absolute ⟩32-Mbyte Disk Read) and 26h (Absolute ⟩32-Mbyte Disk Write) [4]
ENTRY: AL = Drive number (0 = A, 1 = B, etc.)
BX = Pointer to parameter list
CX = −1 (indicates extended (⟩32-Mbyte) format)

RETURN: CF = 0 if successful transfer; CF = 1 if unsuccessful
NOTE: POP AX (error code) on return. Error codes the same as above.
Parameter list structure:

```
rba    dd ? ; first sector (32-bits, 0 origin) to
          ; read/write
count  dw ? ; number of sectors to read/write
buffer dd ? ; data buffer
```

Interrupt 27h—Terminate and Stay Resident [1][2][3][4]

ENTRY: CS = Segment address of program's PSP
DX = Address at which next program may be loaded (highest address to stay resident + 1)
RETURN: None
NOTE: Files are not closed after int 27h. Int 21h function 31h is the preferred method of causing a program to terminate and stay resident.

Interrupt 2Fh—Multiplex Interrupt Function Calls [3][4]

ENTRY: AX = Multiplexing program control:

0100h	= Get PRINT installed state
0101h	= Submit file to PRINT
0102h	= Cancel file in PRINT queue
0103h	= Cancel all files in PRINT queue
0104h	= Pause PRINT and return its status
0105h	= End of PRINT status
0200h	= Get ASSIGN installed state
1000h	= Get SHARE installed state
B700h	= Get APPEND installed state

DS:DX = Pointer to submit packet if AX = 0101h (0 + DWORD pointer to ASCIIZ filespec (no wildcards)); or pointer to ASCIIZ filespec to cancel if AX = 0102h
RETURN: If CF = 1, AX = error code; else
If AL = 0FFh, "program" is installed
If AL = 0, "program" not installed; OK to install
If AL = 1, "program" not installed; not OK to install

Interrupt 67h—Expanded Memory Manager (EMS) [2][3][4]

NOTE: Int 67h is used for LIM EMS in all versions of MS-DOS beginning with version 2.0 but is officially reserved for such use only in MS-DOS versions 4.0 and above. All EMS function numbers are placed in AH, and status/error codes are returned in AH. Status/error codes are:

LIM EMS 3.X, 4.0, MS-DOS 4.0, and AQA EEMS 3.X Error Codes

00h	= Successful operation
80h	= Internal error
81h	= Hardware malfunction
83h	= Invalid handle
84h	= Undefined function requested
85h	= No more handles available
86h	= Error in save or restore of mapping context
87h	= More pages requested than physically exist
88h	= More pages requested than currently available
89h	= Zero pages requested
8Ah	= Invalid logical page number
8Bh	= Illegal physical page number
8Ch	= Page-mapping hardware state save area is full
8Dh	= Page-mapping save failed
8Fh	= Undefined subfunction

LIM EMS 4.0, MS-DOS 4.0, and AQA EEMS 3.X Error Codes

90h	= Undefined attribute type
91h	= Feature not supported
92h	= Successful, but a portion of the source region has been overwritten
93h	= Length of source or destination region exceeds length of region allocated to either source or destination handle
94h	= Conventional and expanded memory regions overlap
95h	= Offset within logical page exceeds size of logical page
96h	= Region length exceeds 1 megabyte
97h	= Source and destination EMS regions have same handle and overlap
98h	= Memory source or destination type undefined
9Ah	= Specified alternate map register set not supported
9Bh	= All alternate map register sets currently allocated
9Ch	= Alternate map register sets not supported
9Dh	= Undefined or unallocated alternate map register set
9Eh	= Dedicated DMA channels not supported
9Fh	= Specified dedicated DMA channel not supported
A0h	= No such handle name
A1h	= Duplicate handle name
A2h	= Attempted to wrap around 1-megabyte conventional address space
A3h	= Contents of source array corrupted or count of mappable segments exceeds total number of mappable segments in system
A4h	= Access denied by operating system

AH = 40h Get Manager Status
ENTRY: None
RETURN: None (status/error code returned in AH)
NOTE: Use only after establishing that EMS driver is present.

AH = 41h Get Page Frame Segment Address
ENTRY: None
RETURN: BX = Segment address of page frame

AH = 42h Get Unallocated Page Count
ENTRY: None
RETURN: BX = Number of unallocated pages
CX = Total number of pages

AH = 43h Allocate Pages
ENTRY: BX = Number of logical pages to allocate
RETURN: DX = Handle

AH = 44h Map/Unmap Handle Pages
ENTRY: AL = Physical page number
BX = Logical page number, or −1 to unmap page
DX = Handle
RETURN: None

AH = 45h Deallocate Pages
ENTRY: DX = Handle
RETURN: None

AH = 46h Get Version
ENTRY: None
RETURN: AL = Version number in BCD

AH = 47h Save Page Map
ENTRY: DX = Handle
RETURN: None

AH = 48h Restore Page Map
ENTRY: DX = Handle
RETURN: None

AH = 4Bh Get Handle Count
ENTRY: None
RETURN: BX = Number of handles

AH = 4Ch Get Handle Pages
ENTRY: DX = Handle
RETURN: BX = Number of logical pages allocated to specified handle

AH = 4Dh Get All Handle Pages
ENTRY: ES:DI = Pointer to handle page array
RETURN: BX = Number of handles in use

AX = 4E00h Get Page Map
ENTRY: ES:DI = Pointer to page map array
RETURN: EMM mapping state stored in page map array pointed to by ES:DI

AX = 4E01h Set Page Map
ENTRY: DS:SI = Pointer to page map array
RETURN: EMM mapping state set from page map array

AX = 4E02h Get and Set Page Map
ENTRY: ES:DI = Pointer to destination page map array
DS:SI = Pointer to source page map array
RETURN: EMM mapping state set from source page map array (DS:SI). Destination page map array (ES:DI) updated with EMM mapping state.

AX = 4E03h Get Size of Page Map Array
ENTRY: None
RETURN: AL = Number of bytes required for source or destination page map array

AX = 4F00h Get Partial Page Map [EMS 4.0]
ENTRY: DS:SI = Pointer to mappable segment array
ES:DI = Pointer to destination partial page map array
RETURN: Partial EMM page map state is contained in destination partial page map array (ES:DI).

AX = 4F01h Set Partial Page Map [EMS 4.0]
ENTRY: DS:SI = Pointer to source partial page map array
RETURN: Partial EMM page map state is updated from source partial page map array (DS:SI)

AX = 4F02h Get Size of Partial Page Map Array [EMS 4.0]
ENTRY: BX = Number of pages in partial page map array
RETURN: AL = Number of bytes required to store partial page map array

AH = 50h Map/Unmap Multiple Handle Pages by Page Number [EMS 4.0]
ENTRY: AL = Subfunction:
00h = physical page specified as page number
01h = physical page specified by segment address
DX = Handle
CX = Number of entries in logical-to-physical map array
DS:SI = Pointer to logical-to-physical map array
RETURN: AH = status/error code

AH = 51h Reallocate Pages [EMS 4.0]
ENTRY: DX = Handle
BX = Number of pages to be allocated to handle
RETURN: BX = Actual number of pages allocated to handle

AX = 5200h Get Handle Attribute [EMS 4.0]
ENTRY: DX = Handle
RETURN: AL = 00h if handle attribute is volatile; AL = 01h if not

AX = 5201h Set Handle Attribute [EMS 4.0]
ENTRY: DX = Handle
BL = 00h if new handle attribute is volatile; BL = 01h if not
RETURN: None

AX = 5202h Get Attribute Capability [EMS 4.0]
ENTRY: None
RETURN: AL = 00h if attribute nonvolatility is supported; AL = 01h if not

AX = 5300h Get Handle Name [EMS 4.0]
ENTRY: DX = Handle
ES:DI = Pointer to 8-character handle name destination buffer
RETURN: Handle name is returned in buffer pointed to by ES:DI

AX = 5301h Set Handle Name [EMS 4.0]
ENTRY: DX = Handle
ES:DI = Pointer to 8-character handle name source buffer
RETURN: Handle name is set based on name in buffer pointed to by ES:DI

AX = 5400h Get Handle Directory [EMS 4.0]
ENTRY: ES:DI = Pointer to handle directory array
RETURN: AL = Number of entries in handle directory

AX = 5401h Search for Named Handle [EMS 4.0]
ENTRY: DS:SI = Pointer to 8-character handle name search buffer
RETURN: DX = Value of named handle

AX = 5402h Get Total Handles [EMS 4.0]
ENTRY: None
RETURN: BX = Total number of handles supported

AH = 55h Alter Page Map and Jump [EMS 4.0]
ENTRY: AL = Subfunction:
00h = physical pages specified as page number
01h = physical pages specified by segment address
DX = Handle
DS:SI = Pointer to map and jump structure
RETURN: Positioned at target address (if AH = 00h)

AH = 56h Alter Page Map and Call [EMS 4.0]
ENTRY: AL = Subfunction:
00h = physical pages specified as page number
01h = physical pages specified by segment address
DX = Handle
DS:SI = Pointer to map and call structure
RETURN: Target address is called (if AH = 00h)
NOTE: Use RETF to return from called location and restore mapping context.

AX = 5602h Page Map Stack Space Size [EMS 4.0]
ENTRY: None
RETURN: BX = Number of stack space bytes required by Alter Page Map and Call function

AH = 57h Move/Exchange Memory Region [EMS 4.0]
ENTRY: AL = Subfunction:
00h = move memory region
01h = exchange memory region
DS:SI = Pointer to source/destination region descriptor
RETURN: None

AX = 5800h Get Mappable Physical Address Array [EMS 4.0]
ENTRY: ES:DI = Pointer to mappable physical address array
RETURN: CX = Number of entries in mappable physical address array

AX = 5801h Get Physical Address Array Entry Count [EMS 4.0]
ENTRY: None
RETURN: CX = Number of entries in mappable physical address array

AX = 5900h Get Hardware Configuration Array [EMS 4.0]
ENTRY: ES:DI = Pointer to hardware configuration array
RETURN: Hardware data is copied into hardware configuration array (pointed to by ES:DI)

AX = 5901h Get Unallocated Raw Page Count [EMS 4.0]
ENTRY: None
RETURN: BX = Number of unallocated raw pages
DX = Total number of raw pages

AH = 5Ah Allocate Standard/Raw Pages [EMS 4.0]
ENTRY: AL = Subfunction
00h = allocate standard pages
01h = allocate raw pages
BX = Number of pages to allocate
RETURN: DX = Handle

AX = 5B00h Get Alternate Map Register Set [EMS 4.0]
ENTRY: None
RETURN: If BL = 0, ES:DI points to map register context save area
If BL 〈 〉 0, BL = pointer to active alternate map register set

AX = 5B01h Set Alternate Map Register Set [EMS 4.0]
ENTRY: If BL = 00h, ES:DI = pointer to map register context save area
If BL 〈 〉 00h, BL = alternate map register set number
RETURN: None

AX = 5B02h Get Alternate Map Save Area Size [EMS 4.0]
ENTRY: None
RETURN: DX = Number of bytes in map register context save area

AX = 5B03h Allocate Alternate Map Register Set [EMS 4.0]
ENTRY: None
RETURN: If BL = 00h, no alternate map register sets are available
If BL 〈 〉 00h, then BL = alternate map register set number allocated

AX = 5B04h Deallocate Alternate Map Register Set [EMS 4.0]
ENTRY: BL = Alternate map register set number
RETURN: None

AX = 5B05h Allocate DMA Register Set [EMS 4.0]
ENTRY: None
RETURN: If BL = 00h, DMA register sets are not supported
If BL 〈 〉 00h, BL = allocated DMA register set number

AX = 5B06h Enable DMA on Alternate Map Register Set [EMS 4.0]
ENTRY: BL = DMA register set number
DL = DMA channel number
RETURN: None

AX = 5B07h Disable DMA on Alternate Map Register Set [EMS 4.0]
ENTRY: BL = DMA register set number
RETURN: None

AX = 5B08h Deallocate DMA Register Set [EMS 4.0]
ENTRY: BL = DMA register set number
RETURN: None

AH = 5Ch Prepare for Warm Boot [EMS 4.0]
ENTRY: None
RETURN: None

AH = 5Dh Enable/Disable OS/E Function Set [EMS 4.0]
ENTRY: AL = Subfunction
00h = enable OS/E function set
01h = disable OS/E function set
02h = return access key
BX, CX = Access key (required only on subsequent calls)
RETURN: BX, CX = Access key returned only on first call of subfunction 00h or 01h