INT 21h - The general function despatcher

Most of the general functions and services offered by DOS are implemented through this interrupt . The functions available are well standardised and should be common to all MSDOS, PCDOS and DOS Plus systems. Well behaved programs, therefore, should use these facilities in preference to any other methods available for the widest range of compatibility.

INT 21h in the 512's implementation of DOS Plus 2.1 provides 77 official functions, two of which are non-functional and return with no action. Within this range some calls have subfunctions which further extend the range of operations.

In all calls, on entry AH defines the function. Other parameters may also be required in other registers. Where a memory block is used by the call this is specified in the normal segment:offset form. In all cases the general programming technique is to set AH to the function pointer, set up the required register contents (and the memory block if necessary) then to issue the call by the assembly code INT instruction. To call the recommended program terminate routine, INT 21h function 4Ch, the relevant parts of the code would be:

```
; Constant equates
Prog_exit               equ 4ch
Function_despatcher     equ 21h
                        org 0100
                        program code here
                        .......
                        .......
                        .......
exit:                                                 ; common program exit
point                   mov     al, Return_code       ; set up result
                        mov     ah, Prog_exit         ; Set up terminate
                                                      ; process
                        int     Function_despatcher   ; and leave
; variable data
Return_code             db ?                          ; Default 0 success
                                                      ; set value on failure
END
```

There are other methods of implementing INT calls, but they are not recommended as normal techniques and are less efficient. The two most likely to be encountered are included here only for infomation.

1. Setting up the entry conditions and executing a long call to 0050h in the PSP (only works in DOS v.2+).
2. Loading the CL register with the function number and executing an intra-sgment call to offset 0050h in the PSP, which contains a long call to the function despatcher. This method only works for function calls of 24h or less, and has the further disadvantage that the contents of register AX are always destroyed.

If calls are made by the approved method the contents of all registers are preserved through calls, except where those registers are used to return results. The obvious exception to this is function

4Bh, EXEC, which transfers control to a spawned program, when the state of all registers except the instruction pointers, but including the stack pointers, should be treated as undefined unless specific returned values are expected.

If spawning is employed register contents which must be be preserved should be pushed onto the stack, and the stack registers themselves (i.e. SS:SI) should be saved in known memory locations for explicit later retrieval.

INT 21h functions 00h to 24h are based on and are, with a few exceptions, direct equivalents to the corresponding CP/M calls. In these calls success or failure is typically signalled by the value returned in register AL. For the remaining (i.e. MSDOS) calls, the carry flag is more usually used, carry clear indicating success, carry set indicating failure of the function, often accompanied by an error code in register AX.

Functions up to and including 57h are documented in this section, all INT 21h functions with a higher number applying to later versions of DOS than 2.11. Note that functions 32h, 34h and 50h and above are included, though they are not supported by DOS Plus, because these do occur in MSDOS version 2.0 and above, and might be encountered in MSDOS v2.0 programs.

Function 0- Program terminate

Action:	On execution the call restores vectors for INTS 22h to 24h from the PSP, flushes any buffers and transfers control to the terminate handler address.

On entry:	AH = 0
	CS = Segment address of PSP

Returns:	Nothing
Notes:	Equivalent of CP/M BDOS call 00h. INT 21h function 4Ch is preferred.

Function 1- Character input with echo

Action:	Reads a character from the standard input device and echoes it to the standard output device. If no character is ready it waits until one is available. I/O can be re-directed, but prevents detection of OEF.
On entry:	AH = 01h
Returns:	AL = 8 bit data input
Notes:	Equivalent to CP/M BDOS call 01h, except that if the character is CTRL-C an INT 23h is performed.

Function 2 - Character output

Action: Outputs a character to the standard output device. I/O can be re-directed, but prevents detection of 'disc full'.

On entry: AH = 02h
DL = 8 bit data (usually ASCII character)

Returns: Nothing

Notes:

Function 3- Auxiliary input

Action: Reads a character from the current auxilliary device.

On entry: AH = 03h

Returns: AL = 8 bit data input

Notes: There is no way to read the status of the serial port or to detect errors through this call, therefore most PC comms packages drive the hardware directly, hence their general incompatibility with the 512.

Function 4- Auxiliary output

Action: Outputs a character to the current auxiliary device.

On entry: AH = 04h
DL = 8 bit data

Returns: Nothing

Notes: There is no way to read the status of the serial port or to detect errors through this call. Comments as Function 3.

Function 5- Printer output

Action: Sends a Character to the current listing device.

On entry: AH = 05h
DL = 8 bit data

Returns: Nothing

Notes: If the printer is busy this call will wait until the data is sent.
There is no way to poll the printer status in DOS.

Function 6- Direct console I/O

Action: Reads a character from the standard input device or returns zero if no character available. Also can write a character to the current standard output device. I/O can be redirected but prevents detection of EOF on input or 'disc full' on output.

On entry: AH = 06h
DL = function requested: 0Ch to 0FEh = output
(DL = character to be output)
0FFh = Input request

Returns: If output - nothing
If input - data ready: zero flag clear, AL = 8 bit data
If data not ready: zero flag set

Notes: This call ignores CTRL-X.

Function 7 - Unfiltered character input no echo

Action: Reads a character from the standard input device without echoing it to the display. If no character is ready it waits until one is available.

On entry: AH = 07h

Returns: AL = 8 bit data input

Notes: This call ignores CTRL-C, use function 8 if CTRL-C processing is required. There is no CP/M equivalent.

Function 08- Character input with no echo

Action: Reads a character from the standard input device without copying it to the display.

If no character is ready it waits until one is available.

On entry: AH = 08h

Returns: AL = 8 bit data input

Notes: If CTRL-C is detected INT 23h is executed.

Function 09- Output character string

Action: Writes a string to the display.

On entry: AH = 09h
DS:DX = segment:offset of string

Returns: Nothing

Notes: The string must be terminated by the $ character (24h), which is not transmitted. Any ASCII codes can be embedded within the string.

Function 0Ah - Buffered input

Action: Reads a string from the current input device up to and including an ASCII carriage return (0Dh), placing the received data in a user-defined buffer Input can be re directed, but this prevents detection of EOF

On entry: AH = 0Ah
DS:DX = segment:offset of string buffer

Returns: Nothing

Notes: The first byte of the buffer specifies the maximum number of characters it can hold (1 to 255). This value must be supplied by the user. The second byte of the buffer is set by DOS to the number of characters actually read, excluding the terminating RETURN. If the buffer fills to one less than its maximum size the bell is sounded and subsequent input is ignored.

If a CTRL-C is detected an INT 23h is executed. Normal DOS keyboard editing is supported during input

Function 0Bh - Get input status

Action: Checks whether a character is available from the standard input device. Input can be redirected

On entry: AH = 0Bh

Returns: AL = 0 if no character available
AL = 0FFh if character available

Notes: Notes: if an input character is waiting this function continues to return a true flag until the character is read by a call to function 1, 6, 7, 8 or 0Ah.

Function 0Ch - Reset input buffer and input

Action: Clears the standard input buffer then invokes one of the standard input functions.

On entry: AH = 0Ch
AL = number of input function to be invoked, which must be 1, 6, 7, 8 or 0Ah.

Returns: If function 0Ah - Nothing
If function 1, 6, 7, or 8 then AL = 8 bit data

Notes: The purpose of this function is to ignore any type-ahead data and force a wait for new character input which must occur after the call is issued.

Function 0Dh - Disc reset

Action: Flush all outstanding file buffers by physically updating to disc.

On entry: AH = 0Dh

Returns: Nothing

Notes: This call does *not* update disc directories for any open files.
If the program fails to close files before the disc is removed and the files have changed size, their directory entries will be incorrect.

Function 0Eh - Set default disc drive

Action: Sets the specified drive to be the default drive and returns the total number of logical drives in the system.

On entry: AH = 0Eh
DL = drive code (A: = 0, B: = 1, etc)

Returns: AL = the number of logical drives in the system.

Notes: In the 512's DOS Plus (2.1) this call always returns five as the number of logical drives, though a maxirnum of four are supported.

Function 0Fh - Open a file

Action: Opens a file and makes it available for read/write operations.

On entry: AH = 0Fh
DS:DX = segment:offset of the file control block (FCB)

Returns: AL = 0 if file found
AL = FFh if file not found

Notes: This call requires a user allocated memory control block.
If the call is successful the FCB data is filled by DOS.

Function 10h - Close file

Action: Closes a file and updates the directory if the file has been modified.

On entry: AH = 10h
DS:DX = Segment offset of the FCB for the file

Returns: AL = 0 if directory updated successfully
AL = 0FFh if fiIe not found in directory

Notes: This call may only be used after a file has been successfully opened and an FCB created.

Function 11h - Find first file

Action: Search for a specified filename in the current directory of the current drive.

On entry: AH = 11h
DS:DX = Segment:offset of the FCB

Returns: AL = 0 if matching filename found
AL = 0FFh if no matching file found

Notes: It is important to use INT 21h function 1Ah to set the DTA to a buffer of adequate size before using this call.

In MSDOS v2 only the '?' wildcard is permitted. If wildcards are specified the first matching name is returned.

If an extended FCB is used, an attribute byte determines the type of files searched for. INT 21h function 4Eh is preferred to this call.

Function 12h - Find next file

Action: Searches the current directory in the current drive for the next matching filename after a previously successful call to function 11h.

On entry: DS:DX = segment:offset of the FCB

Returns: AL = 0 if matching filename found
AL = 0FFh if no matching file found

Notes: As for Function 11h. Function 4Fh is preferred.

Function 13h - Delete file

Action: Deletes all matching files from the current directory.

On entry: AH = 13h
DS:DX = Segment:offset of the FCB

Returns: AL = 0 if matching fie(s) deleted
AL = 0FFh if no matching file found or all matching files are read-only.

Notes: The '?' wildcard is permitted. If more than one match occurs all matched filenames will be deleted.

Function 14h - Sequential read

Action: Reads the next sequential block of data from a file and increments the file pointer.

On entry: AH = 14h
DS.DX = Segment:offset of previously opened FCB

Returns: AL = 0 if successful
AL = 1 if end of file reached
AL = 2 if segment wrap occurs
AL = 3 if partial record read at end of file

Notes: The record is read into memory at the DTA address specified by the most recent call to function 1Ah. The size of the block read is is specified by the record size field in the FCB.

Function 15h - Sequential write

Action: Writes the next sequential block of data to a file and increments the file pointer.

On entry: AH = 15h
DS.DX = Segment:offset of previously opened FCB

Returns: AL = 0 if successful
AL = 1 if disc full
AL = 2 if segment wrap occurs

Notes: The record is written from memory at the DTA address specified by the most recent call to function 1Ah. The size of the block written is specified by the record size field in the FCB.

Function 16h - Create or truncate file

Action: Creates a new entry in the current directory for the named file, or truncates the length of an existing file of the given name to zero length. The file is opened for read/wnte access.

On entry: AH = 16h
DS:DX = segment:offset of unopened FCB

Returns: AL = 00h if successful
AL = 0FFh if unsuccessful (directory full)

Notes: Notes: This call is the equivalent of 'OPENOUT' in BBC BASIC and should be used with care. By using an extended PCB and setting the appropriate bit the opened file may be assigned an attribute. To create files in other directories use function 3Ch.

Function 17h- Rename file

Action: Renames all matching files in the current directory

On entry: AH = 17h
DS:DX = Segment:offset of special FCB

Returns: AL = 0 if successful
AL = 0FFh if no match found or new filename already exists

Notes:

Function 18h - Reserved
Function 19h - Get default disc drive

Action:	Returns the drive code of the current or default drive.
On entry:	AH = 19h
Returns:	AL = drive code (0 = A:, 1 = B: etc)
Notes:	Some DOS functions use drive codes starting at 1 (e.g. function 1Ch) reserving zero for the current dnve.

Function 1Ah - Set disc transfer area address

Action:	Specifies the memory area to be used for subsequent FCB operations.
On entry:	AH = 1Ah DS:DX = Segment:offset of DTA
Returns:	Nothing
Notes:	If this function is not used by a program, the DTA defaults to a 128 byte buffer in the PSP at 080h, the area used to hold the original command tail, which will then be destroyed by any disc activity. It is the programmer's responsibility to ensure that the DTA is large enough for any relevant disc operation. The only exception is that DOS will abort any transfer which would wrap around within the segment. This function *must* be called before using functions 11h, 12h, 14h or 4Fh, to ensure that DOS has a large enough scratch area when searching directories.

Function 1Bh - Get current drive allocation data

Action:	Obtains selected information about the current disc drive and a pointer to the identification byte of the FAT.
On entry:	AH = 1Bh
Returns:	If successful AL = Number of sectors per cluster DS:BX = Segment:offset of FAT identification byte CX = Size in bytes of physical disc sector DX = Number of clusters for the drive If unsuccessful (invalid drive) AL = 0FFh
Notes:	DS:BX points only to the FAT information byte. To read the contents of the FAT into memory use INT 25h. To obtain infomation about discs other than the default drive use function 1Ch. See also function 36h which returns similar data.

Function 1Ch - Get alloc. data for specified drive

Action: Action: As for Function 1Bh, but any drive can be specified.

On entry: AH = 1Ch
DL = Drive code (NOTE 0 = current, 1 = A:, 2 = B:)

Returns: If successful
AL = Number of sectors per cluster
DS.BX = Segment:offset of FAT identification byte
CX = Size in bytes of physical disc sector
DX = Number of clusters for the specified drive
If unsuccessful (invalid drive or critical error)
AL = 0FFh

Notes: Except for the ability to specify a drive this call is the equivalent of Function 1Bh.

Functions 1Dh to 20h - Reserved

Function 2lh - Random read

Action: Reads a selected record from an opened file.

On entry: AH = 21h
DS:DX = Segment:offset of previously opened FCB

Returns: AL = 0 if successful
AL = 1 if end of file reached
AL = 2 if segment wrap occurs
AL = 3 if partial record read at end of file

Notes: The record is read into memory at the DTA address spedfied by the most recent call to Function 1Ah. The size of the block read is specified by the record size field in the FCB.

If the size of the DTA and the record are such that segment wrap occurs, the call fails with a return code of 2. If a partial record read occurs the remaining space is padded with zeros. The current file pointer is not advanced after this function.

Function 22h - Random write

Action:

On entry: AH = 22h
DS:DX = Segment:offset of previously opened FCB

Returns: AL = 0 if successful
AL = 1 if disc full
AL = 2 if segment wrap occurs

Notes: The record is written from memory at the DTA address specified by the most recent call to Function 1Ah. The size of the block written is specified by the record size field in the FCB. If the size of the record and the location of the DTA are such that segment wrap occurs, the call fails with a return code of 2.

Function 23h - Get file size in records

Action: Returns the record count of a matching file.

On entry: AH = 23h
DS:DX = Segment:offset of unopened FCB

Returns: AL = 0 if matching file found
AL = 0FFh if no matching file found

Notes: Before using this call you must set an appropriate record size in the FCB's record size field. After the call the random record field is set to the record count of the specified file.

Function 24h - Set random record number

Action: Sets the random record field of an FCB to correspond to the current file position as recorded in the opened FCB.

On entry: AH = 24h
DS:DX = segment:offset of previously opened FCB

Returns: Nothing. The random record field in the FCB is updated

Notes: This function is used to change from sequential to random I/O file access.

Function 25h - Set interrupt vector

Action: Initialises an interrupt vector to point to the supplied address.

On entry: AH = 25h
AL = Interrupt number
DS:DX = segment:offset of new vector address

Returns: Nothing

Notes: This is the approved way to amend interrupt vector contents.

Before changing the contents of a vector, Function 35h should be used to obtain the original entry, which should be re-instated when your code terminates. The only exceptions to this rule are interrupt vectors 22h to 24h, which are automatically restored from the PSP on program termination.

Function 26h - Create program segment prefix

Action: Copies the PSP of the current program to a specified segment address in free memory, then updates the new PSP to make it usable by a new program.

On entry: AH = 26h
DX = Segment for new PSP

Returns: Nothing

Notes: This call has been rendered obsolete by EXEC, Function 4Bh in DOS 2.0 and later and should no longer be used.

Function 27h - Random block read

Action: Reads one or more sequential records from an open file starting at the file's current record position

On entry: AH = 27h
CX = Number of records to be read
DS.DX = Segment:offset of previously opened FCB

Returns: AL = 0 if all requested records read
AL = 1 if end of file
AL = 2 if segment wrap
AL = 3 if partial record read at end of file
CX = Actual number of records read

Notes: The records are read into memory at the DTA address specified by the most recent call to Function 1Ah. The size and number of blocks read is specified by the random record and the record size field in the FCB.

If the size and. location of the DTA and the number of records to be read are such that segment wrap occurs, the call fails with a return code of 2, possibly after reading one or more records. if a partial record read occurs at the end of the file the remaining record space is padded with zeros The random record, current block and current record fields are updated after this function. The call is similar to Function 21h except that multiple blocks are permitted.

Function 28h - Random block write

Action: Write one or more sequential records to an open file starting at the file's current record position.

On entry: AH = 28h
CX = Number of records to be wriflen or zero (see notes)
DS:DX = Segment:offset of previously opened FCB

Returns: AL = 0 if all requested records written
AL = 1 if disc full
AL = 2 if segment wrap
CX = Actual number of words written

Notes: The records are written from memory at the DTA address specified by the most recent call to Function 1Ah.

If the size and location of the DTA and the number of records to be written are such that segment wrap occurs the call fails with a return code of 2, possibly after writing one or more records.

The random record, current block and current record fields are updated after this function. The call is similar to Function 21h except that multiple records may be read.

If the call is executed with zero in CX no data is written, but the file length is set to the value implied by the current random record field and the record size.

This call is similar to function 22h, except that multiple records may be written.

Function 29h - Parse filename

Action: Parses a text string into the various fields of an FCB.

On entry: AH = 29h
AL = flags to control parsing, as follows:
Bit 3: If set, the extension field in an existing FCB will be modified only if an extension is specified in the string being parsed. If clear, the extension field will always be modified. If no extension is specified the FCB extension field will be set to blanks (20h).

Bit 2: if set, the filename field in an existing FCB will be modified only if a filename is specified in the string being parsed. if clear, the filename field will always be modified. If no filename is specified the FCB filename field will be set to blanks (20h).

Bit 1: if set, the drive ID byte in the resulting FCB will be modified only if a drive ID is specified in the string being parsed. if clear, the drive ID will always be modified. If no drive is specified the drive ID in the resulting FCB will be set to zero, (default).

Bit 0: if set, leading separators will be ignored.

DS:SI = Segment:offset of text string
ES:DI = Segment:offset of FCB

Returns: AL = 0 if no global (wildcard) characters encountered
AL = 1 if parsed string contains global characters
AL = 0FFh if drive specifier is invalid
DS:SI = Segment:offset of first character after parsed name
ES:DI = Segment:offset of formatted, unopened FCB

Notes: Permitted separators are: - : . ; , = + TAB and SPACE

The call regards all control characters, the above separators (when trailing) and < > I / " [and [as terninating characters. If no valid filename is present the contents of ES:DI+1 is a blank. If a '*' occurs in an extension, it and all remaining characters in the FCB are set to '?'. This function (and FCBs in general) cannot be used with file specifications which include a path.

Function 2Ah - Get system date

Action: Returns the system day, month and year plus the day of the week.

On entry: AH = 2Ah

Returns: CX = year (1980 to 2099)
DH = month (1 to 12)
DL = day of month(1 to 31)
AL = day number in week (0 to 6 = Sunday to Saturday)

Notes: The format of the registers returned by this call is the same as that required by Function 2Bh. Although shown above as decimal values for clarity, all values are in hex.

Function 2Bh - Set system date

Action: Reset the date held in the system clock

On entry: AH = 2Bh
CX = year (1980 to 2099)
DH = month (1 to 12)
DL = day of month (1 to 31)

Returns: AL = 0 if successful
AL = if invalid date supplied (no change)

Notes: The system time of day is unaffected by this call.

Function 2Ch - Get system time

Action: Returns the time of day as held by the system clock.

On entry: AH = 2Ch

Returns: CH = hour(0 to 23)
CL = minute (0 to 59)
DH = second (0 to 59)
DL = centiseconds (0 to 99)

Notes: The register format returned by this call is the same as that required by Function 2Dh.

Function 2Dh - Set system time

Action:	Sets the time of day held in the system clock.

On entry:	AH = 2Dh
	CH = hour(0 to 23)
	CL = minute (0 to 59)
	DH = second (0 to 59)
	DL = centiseconds (0 to 99)

Returns:	AL = 0 if time reset successfully AL = 0FFh if invalid time supplied (no change)
Notes:	

Function 2Eh - Set verify flag

Action:	Sets or cancels the system read after write verify flag.
On entry:	AH = 2Fh AL = 0 to turn verification off AL = 1 to turn verification on DL should be set to zero
Returns:	Nothing
Notes:	This call is intended to provide increased data integrity by allowing read after write verification on all data written to disc. It is the equivalent to the DOS command VERIFY and, like the manual command, is non-functional in DOS Plus 2.1.

Function 2Fh - Get DTA address

Action:	Returns the segment:offset of the current DTA for read/write operations.
On entry:	AH = 2Fh
Returns:	ES:BX = Segment.offset of current DTA
Notes:	If the DTA has not been explicitly set it defaults to 080h in the PSP.

Function 30h - Get DOS version

Action:	Returns the version number of the Operating System.
On entry:	AH = 30h

Returns:	AL = Major version number (e.g. 2.10 = 2)
	AH = Minor version number (e.g. 2.10 = 0Ah)

Notes: In the 512 this call returns 2.11.

Function 31h - Terminate & stay resident (keep)

Action: Terminate program execution but leave memory allocated.

On entry: AH = 31h
AL = Return code
DX = memory size to be reserved (in paragraphs)

Returns: Nothing

Notes: Notes: TSR programs are usually re-entered by having previously re-directed an interrupt vector to point back into the code. In this way the program may be re-entered as a result of normal system activity, or as a result of an explicit call by another program.

Function 32h - Get disc info (Undocumented call)

Action: Returns the pointer to the specified disc drive information block

On entry: AH = 32h
DL = drive number (0 = default, 1 = A: etc)

Returns: AL = 0 if drive valid
DS:BX = segment:offset of disk information block
AL = 0FFh if invalid drive number

Notes: This call is unofficial and is not supported by DOS Plus.

Function 33h Get or set CTRL-BREAK flag

Action: Returns or sets the CTRL-BREAK action

On entry: AH = 33h
If getting the status of the flag: AL = 0
If setting the flag: AL = 1
DL = 0 to turn CTRL-BREAK checking off
DL = 1 to turn CTRL-BREAK checking on

Returns: DL = 0 if CTRL-BREAK checking off
DL = 1 if CTRL-BREAK checking on

Notes: Notes: This command is the functional equivalent of the DOS command BREAK. Like that command, in the 512 this call is non-functional.

Function 34h - Find active byte (Undocumented)

Action: Returns the number of currently active processes

On entry: AH = 34h

Returns: ES:BX = Segment:offset of active byte address

Notes: This call is unofficial and is not supported by DOS Plus. In MSDOS it is mainly used by TRS.

Function 35h - Get interrupt vector

Action: Returns the segment:offset of a nominated vector.

On entry: AH = 35h
AL = interrupt number

Returns: ES:BX = Segment offset of interrupt vector contents

Notes: This is the approved way to read interrupt vector contents. The original contents of the vector, after storage, can be amended by a call to function 25h.

Function 36h - Get free disc space

Action: Gives the number of free clusters on a specified disc.

On entry: AH = 36h
DL = Drive code (0 = default, 1 = A: etc)

Returns: If specified drive valid:
AX = Sectors per cluster
BX = number of available clusters
CX = bytes per sector
DX = Clusters (allocation units) per drive
If specified drive is invalid:
AX = 0FFFFh

Notes: Similar information is returned by functions 1Bh and 10h

Function 37h- Reserved

Function 38h - Get country information

Action: Reading geographically variable system constants.

On entry: AH = 38h
AL = 0
DS.DX = Segment:offset of a 32 byte control block

Returns: If unsuccessful Carry flag set AX = Error code

If successful the 32 byte block contents are as follows.

5 byte currency symbol add, null terminated
2 byte thousands separator, null terminated
2 byte decimal separator, null terminated
2 byte date separator, null terminated

1 byte bit field currency format
Bit 0: clear if currency symbol precedes value, set if value precedes currency symbol
Bit 1: clear if no space between the value and the currency symbol, Set if a space required

1 byte time format
Bit 0: clear for 12 hour clock, set for 24 hour clock

2 words for case map call address
2 bytes data list separator, null terminated
5 words reserved

Notes: Unlike its MSDOS counterpart, this call does not permit the stored information to be amended.

Function 39h - Create subdirectory

Action: Creates a new subdirectory using the specified drive and path data.

On entry: AH = 39h
DS:DX = Segment:offset of ASCIIZ path specification

Returns: Carry: clear if successful, set if unsuccessful, when:
AX = 3 if path not found, 5 if access denied

Notes: The function fails if:

1. Any part of the pathname does not exist.
2. A directory of the same name already exists in the same path.
3. The parent directory is the root and it is full.

Function 3Ah - Delete subdirectory

Action: Deletes a specified subdirectory.

On entry: AH = 3Ah
DS:DX = Segment:offset of ASCIIZ path specification

Returns: Carry clear if succesful
Carry set if unsuccessful, when:
AX = Error code as follows:

3: path not found
5: access denied
6: current directory
16: directory contains files

Notes: The function fails if:

1. Any part of the pathname does not exist.
2. The specified directory is the curmnt directory.
3. The specified directory still contains files.

Function 3Bh - Set current directory

Action: Sets the specified directory to be the current directory.

On entry: AH = 3Bh
DS.DX = Segment:offset of ASCIIZ path specification

Returns: Carry clear if successful,
Set if unsuccessful, when:
AX = Error code 3: path not found

Notes: The call fails if any part of the pathname does not exist.

Commonly the current directory is ascertained by a call to function 47h, then stored by a program so the original current directory can be reset on completion of operations.

Function 3Ch - Create or truncate file

Action: Creates a new entry in the specified directory on the specified drive for the named file, or truncates the length of an existing file of the given name and path specification to zero length. The file is opened for read/write access and a 16 bit handle is returned for future access.

On entry: AH = 3Ch
CX = File attribute:
0 = normal
1 = read only,
2 = hidden,
3 = system
DS:DX = Segment:offset of ASCIIZ file specification

Returns: Carry clear if successful: AX = file handle
Carry set if unsuccessful: AX = Error code as follows
3: Path not found

4: No handle available (too many files)
5: Access denied

Notes: The function fails if:

1. Any part of the pathname does not exist.
2. A file of the same name afready exists in the same path with the read only attribute set.
3. The parent directory is the root and it is full.

Function 3Dh - Open file

Action: Opens a file in the specified or default directory on the specified drive for the named file. A 16-bit handle is returned for future access.

On entry: AH = 3Dh
AL = access mode, where:
0 = read access
1 = write access
2 = read/write access
All other bits off
DS.DX = Segment:offset of ASCIIZ file specification

Returns: Carry clear if successful: AX = file handle
Carry set if unsuccessful AX = Error code as follows
2: File not found
3: Path does not exist
4: No handle available (too many files)
5: Access denied
0Ch: Access code invalid

Notes: Any normal system or hidden file with a matching name will be opened by this function. On return the read/write pointer is set to zero, the start of the file.

The call fails if:
1. Any part of the path does not exist.
2. A read only file is opened for write or read/write access.

Function 3Eh - Close file

Action: Closes a successfully opened file. All buffers are flushed to disc and the file handle is freed for re-use. If the file was modified, the date and time stamps and the file length are updated in the directory entry.

On entry: AH = 3Eh
BX = the file handle

Returns: Carry clear if successful, set if failed, when AX = error code 6, invalid handle or not open

Notes: In MSDOS calling this function with a handle of zero closes the standard input device! DOS Plus does not suffer from this bug.

Function 3Fh - Read file or device

Action: Reads a specified number of bytes from a successfully opened file or device.

On entry: AH = 3Fh
BX = File handle
CX = Nurnber of bytes to be read
DS:DX = Segment:offset of buffer area

Returns: Carry clear if successful
AX = number of bytes read
AX = 0 means that EOF was already reached.
Carry set if failed, and AX = Error code as follows:
5: Access denied
6: Invalid handle or not open

Notes: If reading frorn a character device in cooked mode, a maximum of one line will be read, as a carriage return (0Dh) is treated as a record terminator.

If the carry flag is clear and AX is less than CX a partial record was read or there was an error.

Function 40h - Write to file or device

Action: Reads a specified number of bytes from a successfully opened file or device.

On entry: AH = 40h
BX = File handle
CX = Number of bytes to be written
DS:DX = Segment:offset of buffer area

Returns: Carry clear if successful, when
AX = number of bytes written
AX = 0 means the disc is full
Carry set if failed, when:
AX = Error code as follows:
5: Access denied
6: Invalid handle or not open

Notes: If the carry flag is clear and AX is less than CX, this means that a partial record was written or there was an error.

Function 41h - Delete file

Action: Deletes a file from the specified or default disc and directory.

On entry: AH = 41h
DS.DX = Segment:offset of ASCIIZ file specification

Returns: Carry clear if successful, set if failed, when AX = Error code as follows:
2: File not found
5: Access denied

Notes: This deletes a file by deleting its directory entry. The ASCIIZ string specifying the file may not include wildcards. The function fails if:

1. Any part of the path does not exist.
2. The specified file has a read-only attribute.

Function 42h- Move file pointer

Action: Sets the file pointer to the specified position relative to the start or end of the file, or to the current pointer location.

On entry: AH = 42h,
AL = method code as follows.
0: Absolute byte offset from start of the file. (Always +ve double integer)
1: Byte offset from current location (+ve or -ve double integer)
2: Byte offset from the end of the file (+ve or -ve double integer)
BX = File handle
CX = MSB of offset
DX = LSB of offset

Returns: Carry clear if successful
DX = MSB of new pointer location
AX = LSB of new pointer location
Carry set if failed, when AX = Error code as follows:
1: function number invalid
6: invalid handle or not open

Notes: The method code determines the relative base for the pointer move, which uses a 32 bit integer to set the new location.

Method 2, if called with an offset of zero returns the length of the file as the new pointer value.

Method 1 or 2 can set the pointer to a location before the start of the file, but an error will occur if a subsequent attempt is made to use this pointer location.

For all methods (and results) the returned pointer location is always an absolute byte offset from the start of the file.

Function 43h - Get or set file attributes

Action:	Obtains or sets the attributes of the specified file.
On entry:	AH = 43h AL = 0, get file attribute or AL = 1, set file attribute CX = new attribnte (when AL = 1) as follows: bit 5 = archive bit 2 = system bit 1 = hidden bit 0 = read only DS:DX = Segment:offset of ASCIIZ file specification
Returns:	Carry clear if successful, CX = attribute (when AL was 1) Carry set if failed, AX = Error code as follows: 1: function code invalid 2: file not found 3: path not found or file not found 5: attribute cannot be changed
Notes:	This function cannot be used to set a file's volume label bit (3), or the sub-directory bit (4). These may only be changed by using an extended ECB.

Function 44h- Device driver control (IOCTL)

Action:	Passes information directly between an application and a device driver.

On entry:	AH = 44h
	AL = 6 get input status
	AL = 7 - get output status
	BX = handle

Returns:	Carry clear if successful AL = 0: For device or output file = Not Ready. For input file = Pointer at EOF AL = FFh: For device, input or output file = Ready Carry set if failed, when AX = Error code as follows: 1: AL not 6 or 7 5: Acces denied 6: Invalid handle or not open 0Dh: Invalid data
Notes:	This call is a partial implementation of the full MS/PCDOS flindion, as it only supports status checkng in the 512.

Function 45h - Duplicate handle

Action: Returns a second handle for a file or device which has already been successfully opened.

On entry: AH = 45h
BX = existing file or device handle

Returns: Carry clear if successful
AX = new handle
Carry set if failed, when
AX = Error code as follows:
4: No handle available
6: Handle invalid or not open

Notes: If the file pointer attached to one handle is implicitly moved by a seek, read or write, the pointer for the other handle is also moved.

The purpose of this call is to force directory updating for a file without having to close it (and then re-open it). After obtaiidng the new handle, the logical file associated with it is closed by function 3Eh, forcing a directory update, but leaving the original handle available for further input/output operations.

Function 46h - Force duplicate of handle

Action: Given two handles, make the second refer to the same file at the same point as the first.

On entry: AH = 46h
BX = first file handle
CX = second file handle

Returns: Carry clear if successful, set if failed when AX = Error code
4: No handles available
6: Invalid handle or not open

Notes: If the handle specified in CX already refers to an open file, that file is closed before this function is performed.

After the call, if the file pointer attached to one handle is implicitly moved by a seek, read or write, the pointer for the other handle is also moved.

Function 47h - Get current directory

Action: Obtains the ASCIIZ string of the current directory's path.

On entry: AH = 47h
DL = Drive code (0 = default, 1 = A:, etc)
DS:SI = Segment:offset of 64byte scratch buffer

Returns: Carry clear if successful, full directory pathnarne is placed in the buffer.
Carry set if failed when AX = Error code as follows:

4: No handle available
6: Drive specification invalid

Notes: The returned pathnarne does not mdude the drive ID, nor is it prefixed with a '\'. It is terminated by a null byte, therefore if this call is issued from the root directory, the first byte in the buffer will be zero.

Function 48h - Allocate memory

Action: Allocates a block of memory and returns a pointer to the start of the area.

On entry: AH = 48h
BX Number of paragraphs of memory needed

Returns: Carry clear if successful when AX = first segment of allocated block
Carry set if failed when AX = Error code as follows:
7: memory control blocks destroyed
8: insufficient memory, BX = size of largest available block

Notes: If the call succeeds AX:0000 points to the start of the block.

When a COM file loads, it conceptually owns all the remainder of memory from its PSP upwards. This call may be used to lirnit a program's memory allocation to its immediate requirements.

Function 49h - Release memory

Action: Releases memory to make it available to other programs.

On entry: AH = 49h
BX = New requested block size in paragraphs
ES = Start segment of block to be modified

Returns: Carry clear if successful, set if failed when AX = Error code as follows:
7 = memory control blocks destroyed
8 = insufficient memory
9 = incorrect segment in ES
BX = Size of largest available block

Notes: This call modifies the size of a block of memory previously allocated through Function 48h. The call *must* be used by a COM program to release all unused memory before spawning by means of EXEC, Function 4Bh. EXE programs may also use this call.

Function 4Bh - Execute program

Action: Loads a program for execution under the control of an existing program. By means of altering the INT 22h to 24h vectors, the calling prograrn can ensure that, on termination of the called program, control returns to itself.

On entry: AH = 4Bh
AL = 0: Load and execute a program
AL = 3: Load an overlay
DS.DX = segment:offset of the ASCIIZ pathname
ES:BX = Segment:offset of the parameter block
Parameter block bytes:
0-1: Segment pointer to envimmnemnt block
2-3: Offset of command tail
4-5: Segment of command tail
6-7: Offset of the first FCB to be copied to new PSP+5Ch
8-9: Segment of the first FCB
Ah-Bh Offset of the second FCB to be copied to new PSP+6Ch
Ch-Dh Segment of the second FCB

Returns: Carry clear if successful. On return *all* register contents are destroyed, including the stack pointers.
Carry set if failed when AX = Error code as follows:
1: Function invalid
2: File not found or path invalid
5: Access denied
8: Insufficient memory
0Ah: Environment invalid
0Bh Format invalid

Notes: To protect the caller's register contents they should be pushed on the stack and the stack pointers, SS:SP stored in a known location before the call. On return these should be retrieved immediately with interrupts disabled to prevent interrupts occuring before stack integrity is regained.

The ASCIIZ pathnarne must include the drive, path and filename of the program to be loaded. The environment block must be paragraph-aligned and consist of one or more ASCIIZ strings, all terminated by an extra zero byte.

Command tails are in the usual format for PSPs, that is, a count byte and the command tail terminated by a carriage return, which is not included in the count.

All active handles, devices and I/O redirection assignments in the calling program are inherited by the executed program.

Function 4Ch - Terminate program with return code

Action: Terminates execution of a program with return to `COMMAND.COM` or a calling routine, passing back a return code. Allocated memory is freed, vectors for interrupts 22h to 24h are restored from the PSP and all file buffers are flushed to media. All files are closed and directories are updated.

On entry: AH = 4Ch
AL = Return code (Error level)

Returns: Nothing

Notes: This is the approved method of terminating program execution. It is the only way that does not rely on the contents of any segment register and is thus the simplest exit, particularly for EXE files.

The return code can be interrogated by a calling program by means of function 4Dh, and by the batd' file commands, `IF ERRORLEVEL`. Conventionally a return code of zero indicates success, any other value failure. Standard DOS return codes are:

0: Successful operation
1: CTRL-BREAK termination
2: Critical error ternination
3: Terminated and stayed resident

Return code values can be used at the discretion of the programmer (avoiding codes 1 to 3), thus both success or a wide range of failure types may be indicated by varying the code. For the return of result codes to the caller by an EXEced program a better method is to use other registers, but only the contents of register AL are significant to the batch command `ERRORLEVEL`.

Function 4Dh - Get return code

Action: Used by a parent task to obtain the return code of a program executed by a call to function 4Bh.

On entry: AH = 4Dh

Returns: AH = Exit type:
0 Normal termination
1: CTRL-C termination
2: Terminated by critical device error
3: Terminated by a call to function 31h
AL = Return code

Notes: This call is a 'one-shot' function, that is, it will yield the return code of a called program once only.

Function 4Eh - Search for first match

Action:	Searches the default or specified drive:directory for the first occurrence of a matching filename.
On entry:	AH = 4Eh
Returns:	CX = Attribute to use in search DS:DX = Segment:offset of ASCIIZ file specification
Notes:	

CX = 0 if successful. The current DTA is filled as follows:

Bytes
0-20: Reserved for use by DOS in subsequent calls
21: Attribute of matched file
22-23: File time stamp
24-25: File date stamp
26-27: least significant word of file size
28-29: Most significant word of file size
30-42:: Filenarne.extension in ASCIIZ string form
Carry set if failed, AX = Error code as follows
02h path invalid
12h: no rnatching directory entry

This call assumes the DTA has heat set up by a successful call to function 1Ah.

Both wildcards (? and *) are permitted in filenames, but only the first matching name is returned.

if the attribute in CX is zero only normal files are searched. If the volume label attribute bit is set only volume labels are returned. For all other attribute settings, (i.e, hidden, system or directory) those files and normal files are searched

Function 4Fh - Search for next match

Action:	Searches for the next matching file after a previously successful call to Function 4Eh.
On entry:	AH = 4Fh
Returns:	Carry clear if successful The current DTA is filled as follows:- Bytes 0-20 : reserved for use by DOS in subsequent calls 21: Attribute of matched file 22-23: File time stamp 24-25: File date stamp 26-27: least significant word of file size 28-29: Most significant word of file size

30-42: Filenarne.extension in ASCIIZ string form
Carry set if failed, AX = Error code
12h: no matching directory entry

Notes: When used this call requires a DTA containing returned data from a previously successful call to function 4Eh or 4Fh.

Use of function 4Fh is only relevant when the original file spedacation used in function 4Eh included at least one wildcard.

Function 50h - Get disc information (Undocumented call)

Action: Returns a pointer to the disc information block.

On entry: AH = 50h
DL = drive number (0 = default, 1 = A: etc)

Returns: AL = 0 if drive exists
DS:BX = Segment:offset of disc information block
AL = 0FFh if failed

Notes: This call is unofficial and is not supported by DOS Plus.

Function 51h - Reserved

Function 52h- Reserved

Function 53h - Reserved

Function 54h - Get verify flag

Action: Reads the current state of the verify flag.

On entry: AH = 54h

Returns: AL = 0 if verify off
AL = 1 if verify on

Notes: This call is the countepart of function 2Eh. In DOS Plus AL is always returned as zero.

Function 55h - Reserved

Fundion 56h - Rename file

Action: Renames a file and or moves its directory entry to a different directory on the disc.

On entry: AH = 55h
DS:DX = Segment:offset of current ASCIIZ filename ES:DI = Segment:offset of new ASCIIZ filename

Returns: Carry clear if successful
Carry set if failed, AX = Error code as follows:
2: File not found
3 : path not found or the file doesn't exist
5: Access denied
11h : new name not same device

Notes: The call fails if:

1. Any part of the pathnarne does not exist.
2. The new filename refers to a different disc.
3. The new name is in the root directory, which is full.
4. A file with the new path and name already exist.

Function 57h - Get or set file date and time

Action:

On entry: AH = 57h
BX = file handle
AL = 0 to get date and time
AL = 1 to set date and time
CX = time:
bits 9-0Fh = hours (0-23)
bits 5-8 = minutes (0-59)
bits 0-4 = No. of two-second increments (0-29)
DX = date.
bits 9 - = year relative to 1980 (0-119, i.e. 1980-2099)
bits 5-8 = month of year(1 to 12)
bits 0-4 = day of month(1 to 31)

Returns: Carry clear if successful
If getting date and time:
CX = time (in format above)
DX = date (in format above)
Carry set if failed, AX = Error code as follows:
1 - function code invalid
6 - file handle invalid

Notes: The file must have been previously opened or created by a call to function 3Ch or 3Dh.

For simplicity the date and time formats are shown above in the sequence they are stored in the directory.

This completes the list of INT 21h function codes valid in DOS PIus 2.1.

Functions 58h and above are only available in versions of MSDOS later than 2.11, and with the exception of function 63, later than version 3.0.

Interrupts 22h through 24 are not user callable and are therefore not documented. (See chapter 8).