APPENDIX D: DOS INTERRUPT 21H AND 33H LISTING

This appendix lists many of the DOS 21H interrupts, which are used primarily for input, output, and file and memory management. In addition, this appendix covers some functions of INT 33H, the mouse handling interrupt. As was mentioned in Chapter 5, this interrupt is not a part of DOS or BIOS, but is part of the mouse driver software.

SECTION D.1: DOS 21H INTERRUPTS

First, before covering the DOS 21H interrupts, a few notes are given about file management under DOS. There are two commonly used ways to access files in DOS. One is through what is called a file handle, the other is through an FCB, or file control block. These terms are defined in detail below. Function calls 0FH through 28H use FCBs to access files. Function calls 39H through 62H use file handles. Handle calls are more powerful and easier to use. However, FCB calls maintain compatibility down to DOS version 1.10. FCB calls have the further limitation that they reference only the files in the current directory, whereas handle calls reference any file in any directory. FCB calls use the file control block to perform any function on a file. Handle calls use an ASCIIZ string (defined below) to open, create, delete, or rename a file and use a file handle for I/O requests. There are some terms used in the interrupt listing that will be unfamiliar to many readers. DOS manuals provide complete coverage of the details of file managment, but a few key terms are defined below.

ASCIIZ string

This is a string composes of any combination of ASCII characters and terminated with one byte of binary zeros (00H). It is frequently used in DOS 21H interrupt calls to specify a filename or path. The following is an example of an ASCIIZ string that was defined in the data segment of a program:

```
NAME_1    DB 'C:\PROGRAMS\SYSTEM_A\PROGRAM5.ASM',0
```

Directory

DOS keeps track of where files are located by means of a directory. Each disk can be partitioned into one or more directories. The directory listing lists each file in that directory, the number of bytes in the file, the date and time the file was created, and other information that DOS needs to access that file. The familiar DOS command "DIR" lists the directory of the current drive to the monitor.

DTA Disk transfer area

This is essentially a buffer area that DOS will use to hold data for reads or writes performed with FCB function calls. This area can be set up by your program anywhere in the data segment. Function call 1AH tells DOS the location of the DTA. Only one DTA can be active at a time.

FAT File allocation table

Each disk has a file allocation table that gives information about the clusters on a disk. Each disk is divided into sectors, which are grouped into clusters. The size of sectors and clusters varies among the different disk types. For each cluster in the disk, the FAT has a code indicating whether the cluster is being used by a file, is available, is reserved, or has been marked as a bad cluster. DOS uses this information in storing and retrieving files.

FCB File control block

One FCB is associated with each open file. It is composed of 37 bytes of data that give information about a file, such as drive, filename and extension, size of the file in bytes, and date and time it was created. It also stores the current block and record numbers, which serve as pointers into a file when it is being read or written to. DOS INT 21H function calls 0FH through 28H use FCBs to access files. Function 0FH is used to open a file, 16H to create a new file. Function calls 14H - 28H perform read/write functions on the file, and 16H is used to close the file. Typically, the filename information is set up with function call 29H (Parse Filename), and then the address of the FCB is placed in DS:DX and is used to access the file.

File handle

DOS function calls 3CH through 62H use file handles. When a file or device is created or opened with one of these calls, its file handle is returned. The file handle is used thereafter to refer to that file for input, output, closing the file, and so on. DOS has a few predefined file handles that can be used by any Assembly language program. These do not need to be opened before they are used:

Handle value	Refers to
0000	standard input device (typically, the keyboard)
0001	standard output device (typically, the monitor)
0002	standard error output device (typically, the monitor)
0003	standard auxiliary device (AUX1)
0004	standard printer device (PTR1)

PSP Program segment prefix

The PSP is a 256-byte area of memory reserved by DOS for each program. It provides an area to store shared information between the program and DOS.

AH	Function of INT 21H

00 **Terminate the program**

Additional Call Registers
CS = segment address of
PSP (program segment prefix)

Result Registers
None

Note: Files should be closed previously or data may be lost.

01 **Keyboard Input with echo**

Additional Call Registers
None

Result Registers
AL = input character

Note: Checks for ctrl-break.

02 **Output character to monitor**

Additional Call Registers
DL = character to be displayed

Result Registers
None

03 **Asynchronous input from auxiliary device (serial device)**

Additional Call Registers
None

Result Registers
AL = input character

04 **Asynchronous character output**

Additional Call Registers
DL = character to be output

Result Registers
None

05 **Output character to printer**

Additional Call Registers
DL = character to be printed

Result Registers
None

06 **Console I/O**

Additional Call Registers
DL = OFFH if input
or character to be
displayed, if output

Result Registers
AL = 0H if no character available
= character that was input, if
input successful

Note: If input, ZF is cleared and AL will have the character. ZF is set if input and no character was available.

AH	Function of INT 21H

07 **Keyboard input without echo**

Additional Call Registers	Result Registers
None	AL = input character

Note: Does not check for ctrl-break.

08 **Keyboard input without echo**

Additional Call Registers	Result Registers
None	AL = input character

Note: Checks for ctrl-break.

09 **String output**

Additional Call Registers	Result Registers
DS:DX = string address	None

Note: Displays characters beginning at address until a '$' (ASCII 36) is encountered.

0A **String input**

Additional Call Registers	Result Registers
DS:DX = address at which to store string	None

Note: Specify the maximum size of the string in byte 1 of the buffer. DOS will place the actual size of the string in byte 2. The string begins in byte 3.

0B **Get keyboard status**

Additional Call Registers	Result Registers
None	AL = 00 if no character waiting = 0FFH if character waiting

Note: Checks for ctrl-break.

0C **Reset input buffer and call keyboard input function**

Additional Call Registers	Result Registers
AL = keyboard function number 01H, 06H, 07H, 08H or 0AH	None

Note: This function waits until a character is typed in.

AH	Function of INT 21H

0D **Reset disk**

Additional Call Registers	Result Registers
None	None

Note: Flushes DOS file buffers but does not close files.

0E **Set default drive**

Additional Call Registers	Result Registers
DL = code for drive (0=A, 1=B, 2=C, etc.)	AL = number of logical drives in system

0F **Open file**

Additional Call Registers	Result Registers
DS:DX = address of FCB	AL = 00 if successful = 0FFH if file not found

Note: Searches current directory for file. If found, FCB is filled.

10 **Close file**

Additional Call Registers	Result Registers
DS:DX = address of FCB	AL = 00 if successful = 0FFH if file not found

Note: Flushes all buffers. Also updates directory if file has been modified.

11 **Search for first matching filename**

Additional Call Registers	Result Registers
DS:DX = address of FCB	AL = 00 if match is found = 0FFH if no match found

Note: Filenames can contain wildcards '?' and '*'.

12 **Search for next match**

Additional Call Registers	Result Registers
DS:DX = address of FCB	AL = 00 if match found = 0FFH if no match found

Note: This call should be used only if previous call to 11H or 12H has been successful.

AH Function of INT 21H

13 Delete file(s)

Additional Call Registers	Result Registers
DS:DX = address of FCB	AL = 00 if file(s) deleted = 0FFH if no files deleted

Note: Deletes all files in current directory matching filename, provided that they are not read-only. Files should be closed before deleting.

14 Sequential read

Additional Call Registers	Result Registers
DS:DX = address of opened FCB	AL = 00H if read successful = 01H if end of file and no data is read = 02H if DTA is too small to hold the record = 03H if partial record read and end of file is reached

Note: The file pointer, block pointer, and FCB record pointer are updated automatically by DOS.

15 Sequential write

Additional Call Registers	Result Registers
DS:DX = address of opened FCB	AL = 00H if write successful = 01H if disk is full = 02H if DTA is too small to hold the record

Note: The file pointer, block pointer, and FCB record pointer are updated automatically by DOS. The record may not be written physically until a cluster is full or the file is closed.

16 Create/open a file

Additional Call Registers	Result Registers
DS:DX = addr. of unopened FCB = OFFH if unsuccessful	AL = 00H if successful

Note: If the file already exists, it will be truncated to length 0.

17 Rename file(s)

Additional Call Registers	Result Registers
DS:DX = address of FCB	AL = 00H if file(s) renamed = 0FFH if file not found or new name already exists

Note: The old name is in the name position of the FCB; the new name is at the size (offset 16H) position.

AH	Function of INT 21H

18 **Reserved**

19 **Get default drive**

Additional Call Registers
None

Result Registers
AL = 0H for drive A
= 1H for drive B
= 2H for drive C

1A **Specify DTA (disk transfer address)**

Additional Call Registers
DS:DX = DTA

Result Registers
None

Note: Only one DTA can be current at a time. This function must be called before FCB reads, writes, and directory searches.

1B **Get FAT (file allocation table) for default drive**

Additional Call Registers
None

Result Registers
AL = number of sectors per cluster
CX = number of bytes per sector
DX = number of cluster per disk
DS:BX FAT id

1C **Get FAT (file allocation table) for any drive**

Additional Call Registers
DL = drive code
0 for A
1 for B
2 for C

Result Registers
AL = number of sectors per cluster
CX = number of bytes per sector
DX = number of cluster per disk
DS:BX FAT id

1D **Reserved**

1E **Reserved**

1F **Reserved**

20 **Reserved**

21 **Random read**

Additional Call Registers
DS:DX = address of opened FCB

Result Registers
AL = 00H if read successful
= 01H if end of file and no data rea
= 02H if DTA too small for record
= 03H if end of file and partial read

Note: Reads record pointed at by current block and record fields into DTA.

AH Function of INT 21H

22 Random write

Additional Call Registers	Result Registers
DS:DX = address of opened FCB	AL = 00H if write successful
= 01H if disk is full	
= 02H if DTA too small for record	

Note: Writes from DTA to record pointed at by current block and record fields.

23 Get file size

Additional Call Registers	Result Registers
DS:DX = addr. of unopened FCB of records is set in FCB random-record field (offset 0021H)	AL = 00H if file found, number
= 0FFH if no match found	

Note: The FCB should contain the record size before the interrupt.

24 Set random record field

Additional Call Registers	Result Registers
DS:DX = address of opened FCB	None

Note: This sets the random-record field (offset 0021H) in the FCB. It is used prior to switching from sequential to random processing.

25 Set interrupt vector

Additional Call Registers	Result Registers
DS:DX = interrupt handler addr.	None
AL = machine interrupt number	

Note: This is used to change the way the system handles interrupts.

26 Create a new PSP (program segment prefix)

Additional Call Registers	Result Registers
DX = segment addr. of new PSP	None

Note: DOS versions 2.0 and higher recommend not using this service, but using service 4B (exec).

AH Function of INT 21H

27 Random block read

Additional Call Registers
DS:DX = address of opened FCB
CX = number records to be read
= 02H if DTA too small for block
= 03H if EOF and partial block read
CX = number of records actually read

Result Registers
AL = 00H if read successful
= 01H if end of file and no data read

Note: Set the FCB random record and record size fields prior to the interrupt. DOS will update the random record, current block, and current record fields after the read.

28 Random block write

Additional Call Registers
DS:DX = address of opened FCB
CX = number records to write
= 02H if DTA too small for block
CX = number of records actually written

Result Registers
AL = 00H if write successful
= 01H if disk is full

Note: Set the FCB random record and record size fields prior to the interrupt. DOS will update the random record, current block and current record fields after the write. If CX = 0 prior to the interrupt, nothing is written to the file and the file is truncated or extended to the length computed by the random record and record size fields.

29 Parse filename

Additional Call Registers
DS:SI = address of command line
ES:DI = address of FCB
AL = parsing flags in bits 0-3
Bit 0 = 1 if leading separators are to be ignored; otherwise no scan-off takes place
Bit 1 = 1 if drive ID in FCB will be changed only if drive was specified in command line
Bit 2 = 1 if filename will be changed only if filename was specified in command line
Bit 3 = 1 if extension will be changed only if extension was specified in command line

Result Registers
DS:SI = address of first char after
ES:DI = address of first byte of formatted unopened FCB
AL = 00H if no wildcards were in filename or extension
= 01H if wildcard found
= 0FFH if drive specifier is invalid

Note: The command line is parsed for a filename, then an unopened FCB is created at DS:SI. The command should not be used if path names are specified.

AH Function of INT 21H

2A Get system date

Additional Call Registers	Result Registers
None	CX = year (1980-2099) DH = month (1-12) DL = day (1-31) AL = day of week code (0 = Sunday, ... , 6 = Saturd

2B Set system date

Additional Call Registers	Result Registers
CX = year (1980-2099) DH = month (1-12) DL = day (1-31)	AL = 00H if date set = 0FFH if date not valid

2C Get system time

Additional Call Registers	Result Registers
None	CH = hour (0 .. 23) CL = minute (0 .. 59) DH = second (0 .. 59) DL = hundredth of second (0 .. 99)

Note: The format returned can be used in calculations but can be converted to a printable format.

2D Set system time

Additional Call Registers	Result Registers
CH = hour (0 .. 23) CL = minute DH = second DL = hundredth of second	AL = 00H if time set = 0FFH if time invalid

2E Set/reset verify switch

Additional Call Registers	Result Registers
AL = 0 to turn verify off = 1 to turn verify on	None

Note: If verify is on, DOS will perform a verify every time data is written to disk. An interrupt call to 54H gets the setting of the verify switch.

2F Get DTA (disk transfer area)

Additional Call Registers	Result Registers
None	ES:BX = address of DTA

30 Get DOS version number

Additional Call Registers	Result Registers
None	AL = major version number (0,2,3,etc.) AH = minor version number

31 Terminate process and stay resident (KEEP process)

Additional Call Registers	Result Registers
AL = binary return code	None DX = memory size in paragraphs

Note: This interrupt call terminates the current process and attempts to place the memory size in paragraphs in the initial allocation block, but does not release any other allocation blocks. The return code in AL can be retrieved by the parent process using interrupt 21 call 4DH.

32 Reserved

33 Ctrl-break control

Additional Call Registers	Result Registers
AL = 00 to get state of ctrl-break check = 01 to modify state of ctrl-break check DL = 00 to turn check off = 01 to turn check on	DL = 00 if ctrl-break check off = 01 if ctrl-break check on

Note: When ctrl-break check is set to off, DOS minimizes the times it checks for ctrl-break input. When it is set to on, DOS checks for ctrl-break on most operations.

34 Reserved

35 Get interrupt vector address

Additional Call Registers	Result Registers
AL = interrupt number	ES:BX = address of interrupt handler

36 Get free disk space

Additional Call Registers	Result Registers
DL = drive code (0 = default, 1 = A, 2 = B,etc.)	AX = FFFFH if drive code invalid = sectors per cluster if valid BX = number of available clusters CX = bytes per sector DX = total clusters per drive

AH	Function of INT 21H

37 Reserved

38 Country dependent information

Additional Call Registers	Result Registers
DS:DX = address of 32-byte block of memory AL = function code	None

39 Create subdirectory (MKDIR)

Additional Call Registers	Result Registers
DS:DX = address of ASCIIZ path name of new subdirectory AX = 3 if path not found	Carry flag = 0 if successful = 1 if failed = 5 if access denied

3A Remove subdirectory (RMDIR)

Additional Call Registers	Result Registers
DS:DX = address of ASCIIZ path name of subdirectory AX = 3 if path not found = 5 if directory not empty = 15 if drive invalid	Carry flag = 0 if successful = 1 if failed

Note: The current directory cannot be removed.

3B Change the current subdirectory (CHDIR)

Additional Call Registers	Result Registers
DS:DX = address of ASCIIZ path name of new subdirectory	Carry flag = 0 if successful = 1 if failed AX = 3 if path not found

3C Create a file

Additional Call Registers	Result Registers
DS:DX = address of ASCIIZ path and file name CX = file attribute	Carry flag = 0 if successful = 1 if failed AX = handle if successful = 3 if path not found = 5 if access denied

Note: Creates a new file if filename does not exist, otherwise truncates the file to length zero. Opens the file for reading or writing. A 16-bit handle will be returned in AX if the create was successful.

AH Function of INT 21H

3D Open file

Additional Call Registers
DS:DX = addres of ASCIIZ path and file name
AL = mode flags (see below)

Result Registers
Carry flag = 0 if successful
= 1 if failed
AX = 16-bit file handle if successful
= 1 if function number invalid
= 2 if file not found
= 3 if path not found
= 4 if handle not available
= 5 if access denied
= 0CH if access code invalid

AL mode flag summary:

76543210 (bits)	Result
000	open for read
001	open for write
010	open for read/write
0	reserved
000	give others compatible access
001	read/write access denied to others
010	write access denied to others
011	read access denied to others
100	give full access to others
0	file inherited by child process
1	file private to current process

3E Close file

Additional Call Registers
BX = file handle

Result Registers
Carry flag = 0 if successful
= 1 if failed
AX = 6 if invalid handle or file not open

Note: All internal buffers are flushed before the file is closed.

3F Read from file or device

Additional Call Registers
DS:DX = buffer address
BX = file handle
CX = number of bytes to read

Result Registers
Carry flag = 0 if successful
= 1 if failed
AX = number of bytes actually read,
= 5 if access denied
= 6 if file not open or invalid handle

Note: When reading from the standard device (keyboard), at most one line of text will be read, regardless of the value of CX.

AH	Function of INT 21H

40 **Write to file or device**

Additional Call Registers	Result Registers
DS:DX = buffer address	Carry flag = 0 if successful
BX = file handle	= 1 if failed
CX = number of bytes to write	AX = number of bytes actually written if successful
	= 5 if access denied
	= 6 if file not open or invalid handle

Note: If the carry flag is clear and AX is less than CX, a partial record was written or a disk full or other error was encountered.

41 **Delete file (UNLINK)**

Additional Call Registers	Result Registers
DS:DX = address of ASCIIZ file specification	Carry flag = 0 if successful
	= 1 if failed
	AX = 2 if file not found
	= 5 if access denied

Note: This function cannot be used to delete a file that is read-only. First, change the file's attribute to 0 by using interrupt 21 call 43H, then delete the file. No wildcard characters can be used in the filename. This function works by deleting the directory entry for the file.

42 **Move file pointer (LSEEK)**

Additional Call Registers	Result Registers
BX = file handle	Carry flag = 0 if successful
CX:DX = offset	= 1 if fail
AL = 0 to move pointer offset bytes from start of file	AX = 1 if invalid function number
= 1 to move pointer offset bytes from current location	= 6 if file not open or invalid handle
= 2 to move pointer offset bytes from end-of-file	DX:AX = absolute offset from start of file if successful

Note: To determine file size, call with AL = 2 and offset = 0.

43 **Get or set file mode (CHMOD)**

Additional Call Registers	Result Registers
DS:DX = address of ASCIIZ file specifier	Carry flag = 0 if successful
	= 1 if failed
AL = 0H to get attribute	CX = current attribute if set
= 1H to set attribute	AX = 1 if invalid function number
CX = attribute if setting	= 2 if file not found
= attribute codes if getting (see below)	= 3 if file does not exist or path not found
	= 5 if attribute cannot be changed

AH Function of INT 21H

43 Get or set file mode (CHMOD) (continued from previous page)

76543210	attribute code bits
0	reserved
0	reserved
x	archive
0	directory (do not set with 43H; use extended FCB)
0	volume-label (do not set with 43H; use ext. FCB)
x	system
x	hidden
x	read-only

44 I/O device control (IOCTL)

Additional Call Registers
AL = 00H to get device info
= 01H to set device info
= 02H char read device to buffer
= 03H char write buffer to device
= 04H block read device to buffer
= 05H block write buffer to device
= 06H check input status
= 07H check output status
= 08H test if block device changeable
= 09H test if drive local or remote
= 0AH test if handle local or remote
= 0BH to change sharing retry count
= 0CH char device I/O control
= 0DH block device I/O control
= 0EH get map for logical drive
= 0FH set map for logical drive
DS:DX = data buffer
BX = file handle; CX = number of bytes

Result Registers
AX = number of bytes
transferred if CF=0
otherwise = error code

45 Duplicate a file handle (DUP)

Additional Call Registers
BX = opened file handle

Result Registers
Carry flag = 0 if successful
= 1 if failed
AX = returned handle if successful
= 4 if no handle available
= 6 if handle invalid or not open

Note: The two handles will work in tandem; for example, if the file pointer of one handle is moved, the other will also be moved.

AH Function of INT 21H

46 Force a duplicate of a handle (FORCDUP)

Additional Call Registers	Result Registers
BX = first file handle	Carry flag = 0 if successful
CX = second file handle	= 1 if failed
	AX = 4 if no handles available
	= 6 if handle invalid or not open

Note: If the file referenced by CX is open, it will be closed first. The second file handle will be forced to point identically to the first file handle. The two handles will work in tandem; for example, if the file pointer of one handle is moved, the other will also be moved.

47 Get current directory

Additional Call Registers	Result Registers
DL = drive code	Carry flag = 0 if successful
(0 = default,1 = A,...)	= 1 if failed
DS:SI = address of 64-byte buffer	DS:SI = ASCIIZ path specifier
	AX = OFH if drive specifier invalid

Note: The returned pathname does not include drive information or the leading "\".

48 Allocate memory

Additional Call Registers	Result Registers
BX = number of paragraphs	Carry flag = 0 if successful
	= 1 if failed
	AX = points to block if successful
	= 7 if memory control blocks destroyed
	= 8 if insufficient memory
	BX = size of largest block available if failed

49 Free allocated memory

Additional Call Registers	Result Registers
ES = segment address of block	Carry flag = 0 if successful
being released	= 1 if failed
	AX = 7 if memory control blocks destroyed
	= 9 if invalid memory block addr in ES

Note: Frees memory allocated by 48H.

AH Function of INT 21H

4A Modify memory allocation (SETBLOCK)

Additional Call Registers
ES = segment address of block
BX = requested new block size
in paragraphs

Result Registers
Carry flag = 0 if successful
= 1 if failed
BX = max available block size
if failed
AX = 7 if memory control blocks destroyed
= 8 if insufficient memory
= 9 if invalid memory block
address in ES

Note: Dynamically reduces or expands the memory allocated by a previous call to interrupt 21 function 48H.

4B Load and/or execute program (EXEC)

Additional Call Registers
DS:DX = address of ASCIIZ path
and filename to load
ES:BX = address of
parameter block
AL = 0 to load and execute
= 3 to load, not execute

Result Registers
AX = error code if CF not zero

4C Terminate a process (EXIT)

Additional Call Registers
AL = binary return code

Result Registers
None

Note: Terminates a process, returning control to parent process or to DOS. A return code can be passed back in AL.

4D Get return code of a subprocess (WAIT)

Additional Call Registers
None

Result Registers
AL = return code
AH = 00 if normal termination
= 01 if terminated by ctrl-break
= 02 if terminated by critical
device error
= 03 if terminated by call to
interrupt 21 function 31H

Note: Returns the code sent via interrupt 21 function 4CH. The code can be returned only once.

AH Function of INT 21H

4E Search for first match (FIND FIRST)

Additional Call Registers	Result Registers
DS:DX = address of ASCIIZ file specification	Carry flag = 0 if successful = 1 if failed
CX = attribute to use in search	AX = error code

Note: The filename should contain one or more wildcard characters. Before this call, a previous call to interrupt 21 function 1AH must set the address of the DTA. If a matching filename is found, the current DTA will be filled in as follows:

Bytes 0 - 20: reserved by DOS for use on subsequent search calls
21 : attribute found
22 - 23: file time
24 - 25: file date
26 - 27: file size (least significant word)
28 - 29: file size (most significant word)
30 - 42: ASCIIZ file specification

4F Search for next filename match (FIND NEXT)

Additional Call Registers	Result Registers
None	Carry flag = 0 if successful = 1 if failed AX = error code

Note: The current DTA must be filled in by a previous interrupt 21 4EH or 4FH call. The DTA will be filled in as outlined on interrupt 21 function 4E.

50 Reserved

51 Reserved

52 Reserved

53 Reserved

54 Get verify state

Additional Call Registers	Result Registers
None	AL = 00 if verify OFF = 01 if verify ON

Note: The state of the verify flag is changed via interrupt 21 function 2EH.

55 Reserved

AH Function of INT 21H

56 Rename file

Additional Call Registers
DS:DX = address of old ASCIIZ filename specification
ES:DI = address of new ASCIIZ filename specification

Result Registers
Carry flag = 0 if successful
= 1 if failed
AX = 2 if file not found
= 3 if path or file not found
= 5 if access denied
= 11H if different device in new name

Note: If a drive specification is used, it must be the same in the old and new filename specifications. However, the directory name may be different, allowing a move and rename in one operation.

57 Get/set file date and time

Additional Call Registers
AL = 00 to get
= 01 to set
BX = file handle
CX = time if setting
DX = date if setting

Result Registers
Carry flag = 0 if successful
= 1 if failed
CX = time if getting
DX = date if getting
AX = 1 if function code invalid
= 6 if handle invalid

Note: The file must be open before the interrupt. The format of date and time is:

TIME:
Bits 0BH-0FH hours (0-23)
05H-0AH minutes (0-59)
00H-04H number of 2-second increments (0-29)

DATE:
Bits 09H-0FH year (rel.1980)
05H-08H month (0-12)
00H-04H day (0-31)

58 Get/set allocation strategy

Additional Call Registers
AL = 00 to get strategy
= 01 to set strategy
BX = strategy if setting
00 if first fit
01 if best fit
02 if last fit

Result Registers
Carry flag = 0 if successful
= 1 if failed
AX = strategy if getting
= error code if setting

59 Get extended error information

Additional Call Registers
BX = 00

Result Registers
AX = extended error code
(see Table D-1)
BH = error class
BL = suggested remedy
CH = error locus

Warning! This function destroys the contents of registers CL, DX, SI, DI, BP, DS, and ES. Error codes will change with future version of DOS.

AH Function of INT 21H

5A Create temporary file

Additional Call Registers	Result Registers
DS:DX = address of ASCIIZ path	Carry flag = 0 if successful
CX = file attribute	= 1 if failed
(00 if normal, 01 if read-only,	AX = handle if successful
02 if hidden, 04 if system)	= error code if failed
	DS:DX = address of ASCIIZ path specification if successful

Note: Files created with this interrupt function are not deleted when the program terminates.

5B Create new file

Additional Call Registers	Result Registers
DS:DX = address of ASCIIZ	Carry flag = 0 if successful
file specification	= 1 if failed
CX = file attribute	AX = file handle if successful
00 if normal	= error code if failed
01 if read-only	
02 if hidden	
04 if system	

Note: This function works similarly to interrupt 21 function 3CH; however, this function fails if the file already exists, whereas function 3CH truncates the file to length zero.

5C Control record access

Additional Call Registers	Result Registers
AL = 00 to lock ,= 01 to unlock	Carry flag = 0 if successful
BX = file handle	= 1 if failed
CX:DX = region offset	AX = error code
SI:DI = region length	

Note: Locks or unlocks records in systems that support multitasking or networking.

5D Reserved

AH AL Function of INT 21H

5E 00 Get machine name

Additional Call Registers	Result Registers
DS:DX = address of buffer	Carry flag = 0 if successful = 1 if failed CH = 0 if name undefined ≠ 0 if name defined CL = NETBIOS number if successful DS:DX = address of identifier if successful AX = error code

Note: Returns a 15-byte ASCIIZ string computer identifier.

5E 02 Set printer setup

Additional Call Registers	Result Registers
BX = redirection list index CX = setup strength length DS:SI = address of setup string	Carry flag = 0 if successful = 1 if failed AX = error code

Note: This function specifies a string that will precede all files sent to the network printer from the local node in a LAN. Microsoft Networks must be running in order to use this function.

5E 03 Get printer setup

Additional Call Registers	Result Registers
BX = redirection list index ES:DI = address of buffer	Carry flag = 0 if successful = 1 if failed AX = error code CX = length of setup string ES:DI = setup string if successful

5F 02 Get redirection list

Additional Call Registers	Result Registers
BX = redirection list index DS:SI = address of 16-byte device name buffer ES:DI = address of 128-byte network name buffer	Carry flag = 0 if successful = 1 if failed BH = device status flag bit 1 = 0 if valid device = 1 in invalid device BL = device type CX = parameter value DS:SI = addr. ASCIIZ local device name ES:DI = addr. ASCIIZ network name AX = error flag

5F 03 Redirect device

Additional Call Registers	Result Registers
BL = device type 03 printer 04 drive	Carry flag = 0 if successful = 1 if failed AX = error code CX = caller value DS:SI = address of ASCIIZ local device name ES:DI = address of ASCIIZ network name

Note: Used when operating under a LAN, this function allows you to add devices to the network redirection list.

5F 04 Cancel redirection

Additional Call Registers	Result Registers
DS:SI = address of ASCIIZ local device name	Carry flag = 0 if successful = 1 if fail AX = error code

Note: Used when operating under a LAN, this function allows you to delete devices from the network redirection list.

60 Reserved

61 Reserved

62 Get PSP (program segment prefix) address

Additional Call Registers	Result Registers
None	BX = address of PSP

A summary of the IBM error codes is given in Table D-1.

TABLE D-1: Extended Error Code Information

Code	Error
1	invalid function number
2	file not found
3	path not found
4	too many open files
5	access denied
6	invalid handle
7	memory control blocks destroyed
8	insufficient memory
9	invalid memory block address
10	invalid environment
11	invalid format
12	invalid access code
13	invalid data
14	unknown unit
15	invalid disk drive
16	attempt to remove current directory
17	not same device
18	no more files
19	attempt to write on write-protected diskette
20	unknown unit
21	drive not ready
22	unknown command
23	data error (CRC)
24	bad request structure length
25	seek error
26	unknown media type
27	sector not found
28	printer out of paper
29	write fault
30	read fault
31	general failure
32	sharing violation
33	lock violation
34	invalid disk change
35	FCB unavailable
36	sharing buffer overflow
37-49	reserved
50	network request not supported
51	remote computer not listening
52	duplicate name on network
53	network name not found
54	network busy
55	network device no longer exists
56	net BIOS command limit exceeded
57	network adapter hardware error
58	incorrect response from network
59	unexpected network error
60	incompatible remote adapter
61	print queue full
62	not enough space for print file
63	print file was deleted
64	network name not found
65	access denied
66	network device type incorrect
67	network name not found
68	network name limit exceeded
69	net BIOS session limit exceeded
70	temporarily paused
71	network request not accepted
72	print or disk redirection is paused
73-79	reserved
80	file exists
81	reserved
82	cannot make directory entry
83	fail on INT 24
84	too many redirections
85	duplicate redirection
86	invalid password
87	invalid parameter
88	network device fault

Reference: THE 80X86 IBM PC AND COMPATIBLE COMPUTERS (VOLUMES 1 &2) ASSEMBLY LANGUAGE, DESIGN AND INTERFACING, Muhammed Ali Mazidi, Janice Gillispie Mazidi