

BIOS Interrupts And Functions

The various functions, which the ROM-BIOS makes available for the basic communication between a program and the hardware, can be accessed using interrupts 10H to 1AH. Besides functions for the access to the video-hardware, the keyboard, hard drives and diskette drives, this includes checking configuration data, as well as programming of the serial and parallel interface and the battery-buffered real-time clock.

Here is an overview of the various interrupts and their services. Please note, the various functions of Interrupt 13H are explained twice, separated according to their use in relation to diskette and hard drives. Depending on the need for access to diskettes or hard drives, please consult the proper section.

Unless otherwise stated, the various functions are available on all types of PCs. An entire series of functions have only been available since the introduction of the XT models, others only since the introduction of the AT series. The many BIOS enhancements which companies like Compaq added to the original ROM-BIOS were omitted here, since they did not establish themselves as a standard. This is also valid for the enhanced functions of the PS/2 systems of IBM, which are completely compatible with the functions presented here.

Interrupt 10H, Function 00H	BIOS
Video: Set video mode	

Selects and initializes a video mode and clears the screen. This function is a fast method of clearing the screen while maintaining the current video mode.

Input	AH = 00H
	AL = Video mode
	0: 40x25 text mode, monochrome (color card)
	1: 40x25 text mode, color (color card)
	2: 80x25 text mode, monochrome (mono card)
	3: 80x25 text mode, color (color card)
	4: 320x200 4-color graphics (color card)
	5: 320x200 4-color graphics (color card)
	(colors displayed in monochrome)
	6: 640x200 2-color graphics (color card)
	7: Internal mode (mono card)

Output	No output
---------------	-----------

Remarks:

The colors for modes 4, 5 and 6 can be set with function 11.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 01H	BIOS
Video: Define cursor type	

Defines the starting and ending lines of the cursor. This cursor exists independently of the current screen page.

Input	AH = 01H
	CH = Starting line of the cursor
	CL = Ending line of the cursor

Output	No output
---------------	-----------

Remarks

The values allowed for the cursor's starting and ending line depend on the installed video card. The following values are permitted:

Monochrome display cards:	0-13
Color display cards:	0-7

BIOS defaults to the following values:

Monochrome display cards:	11-12
Color display cards:	6-7

You can use this function to set the cursor only within the permitted ranges. Setting cursor lines outside these parameters may result in an invisible cursor or system problems.

The contents of the BX, CX, DX registers and the segment registers SS, CS and DS are not affected by this function. The contents of all the other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 02H	BIOS
Video: Position cursor	

Repositions the cursor, which determines the screen position for character output by using one of the BIOS functions.

Input	AH = 02H
	BH = Screen page number
	DH = Screen line
	DL = Screen column

Output	No output
---------------	-----------

Remarks

The blinking cursor moves through this function when the addressed screen page is the current screen page.

Values for the screen line parameter range from 0 to 24.

Values for the screen column parameter range from 0 to 79 (for an 80-column display) or from 0 to 39 (for a 40-column display), depending on the selected video mode.

You can make the cursor disappear by moving it to a nonexistent screen position (e.g., column 0, line 25).

The number of the screen page parameter depends on how many screen pages are available to the video card.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 03H**BIOS**

Video: Read cursor position

Senses the text cursor's position, starting line and ending line in a screen page.

Input

AH = 03H

BH = Screen page number

Output

DH = Screen line in which the cursor is located

DL = Screen column in which the cursor is located

CH = Starting line of the blinking cursor

CL = Ending line of the blinking cursor

Remarks

The number of the screen page parameter depends on how many screen pages are available to the video card.

Line and column coordinates are related to the text coordinate system.

The contents of the BX register and the SS, CS and DS segment registers are not affected by this function. The contents of all the other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 04H**BIOS**

Video: Read lightpen position

Senses the position of the lightpen on the screen if applicable.

Input

AH = 04H

Output

AH = 0: Lightpen position unreadable

AH = 1: Lightpen position readable

DH = Screen line of the lightpen (text mode)

DL = Screen column of the lightpen (text mode)

CH = Screen line of the lightpen (graphic mode)

BX = Screen column of the lightpen (graphic mode)

Remarks:

This function call must be repeated until 1 is returned in the AH register, because only then can coordinates be read from the other registers.

Coordinates indicated represent the current video mode's resolution.

Usually the coordinates of the light pen cannot be accurately sensed in the graphic mode. The Y-coordinate (line) is always a multiple of 2, so it isn't possible to determine whether the lightpen is in line 8 or 9. The X-coordinate (column) is always a multiple of 4 in 320x200 graphic mode and a multiple of 8 in the 640x200 bitmap mode.

The contents of the CL register and the SS, CS and DS segment registers are not affected by this function. The contents of all the other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 05H	BIOS
Video: Select current screen page	

Selects the current screen page (text mode only) which should be displayed.

Input

AH = 05H

AL = Screen page number

Output

No output

Remarks

The number of the screen page depends on the number of screen pages available to the video card.

On switching to a new screen page, the screen cursor points to the position of the text cursor in this page.

Switching between various screen pages does not affect their contents (the individual characters).

You can write characters to an inactive screen page.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of the other registers, such as the SI and DI registers, may change.

Interrupt 10H, Function 06H	BIOS
Video: Initialize window/scroll text upward	

Clears window or scrolls a portion of the current screen page up by one or more lines, depending on the input.

Input

AH = 06H

AL = Number of window lines to be scrolled upward (0=clear window)

CH = Screen line of the upper left corner of the window

CL = Screen column of the upper left corner of the window

DH = Screen line of the lower right corner of the window

DL = Screen column of the lower right corner of the window

BH = Color (attribute) for blank line(s)

Output No output

Remarks

Initializing a window (placing a 0 in the AL register) fills the window with blank spaces (ASCII code 32).

The contents of the lines scrolled out of the window are lost and cannot be restored.

Function 0 of this interrupt is better for clearing the entire screen.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 07H

BIOS

Video: Initialize window/scroll text downward

Clears window or scrolls a portion of the current screen page up by one or more lines, depending on the input.

Input AH = 07H

AL = Number of window lines to be scrolled downward (0=clear window)

CH = Screen line of the upper left corner of the window

CL = Screen column of the upper left corner of the window

DH = Screen line of the lower right corner of the window

DL = Screen column of the lower right corner of the window

BH = Color (attribute) for blank line(s)

Output No output

Remarks

This function only affects the current screen page.

Initializing a window (placing a 0 in the AL register) fills the window with blank spaces (ASCII code 32).

The contents of the lines scrolled out of the window are lost and cannot be restored.

Function 0 of this interrupt is better for clearing the entire screen.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 08H

BIOS

Video: Read character/attribute

Reads the ASCII code of the character at the current cursor position and its color (attribute).

Input	AH = 08H
	BH = Screen page number
Output	AL = ASCII code of the character
	AH = Color (attribute)

Remarks

The number of the screen page depends on the number of screen pages made available to the video card.

This function can also be called in graphic mode. The function compares the bit pattern of the character on the screen with the bit pattern of the character in character ROM of the video card and with the character patterns stored in a RAM table whose addresses appear in interrupt 1FH. If the character cannot be identified, the AL register contains the value-0 after the function call.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of the other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 09H	BIOS
Video: Write character/attribute	

Writes a character with a certain color (attribute) to the current cursor position in a predefined screen page.

Input	AH = 09H
	BH = Screen page number
	CX = Number of times to write the character
	AL = ASCII code of the character
	BL = Attribute

Output	No output
---------------	-----------

Remarks

If the character should be displayed several times (the value of the CX register is greater than 1), all characters must fit into the current screen line in the graphic mode.

The control codes (e.g., bell, carriage return) appear as normal ASCII codes.

This function can display characters in graphic mode. The patterns of the characters, with the codes from 0 to 127, are determined by a table in ROM. The patterns of the characters with the codes from 128 to 255 are determined by a RAM table that was previously installed by the DOS GRAFTABL command.

In text mode, the contents of the BL register define the attribute byte of the character. In graphic mode this register determines the color of the character. The 640x200 bitmap mode only allows the values 0 and 1 for selecting colors from the color palette. The 320x200 bitmap mode only allows the values 0 to 3 for selecting colors from the color palette.

If the graphic mode is active during character output and bit 7 of the BL register is set, an exclusive OR is performed on the character pattern and the graphic pixels behind the character pattern.

After character output, the cursor remains in the same position as the character.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 0AH	BIOS
Video: Write character	

Writes a character to the current cursor position in a predefined screen page by using the color of the character previously at this position.

Input	AH = 0AH
	BH = Screen page number
	CX = Number of times to write the character
	AL = ASCII code of the character

Output	No output
---------------	-----------

Remarks

If the character should be displayed several times (the value of the CX register is greater than 1), all characters must fit into the current screen line in the graphic mode.

The control codes (e.g., bell, carriage return) appear as normal ASCII codes.

This function can display characters in graphic mode. The patterns of the characters with the codes from 0 to 127 are determined by a table in ROM and the patterns of the characters with the codes from 128 to 255 are determined by a RAM table previously installed by the GRAFTABL command.

In text mode, the contents of the BL register define the attribute byte of the character. In graphic mode this register determines the color of the character. The 640x200 bitmap mode only allows the values 0 and 1 for selecting colors from the color palette. The 320x200 bitmap mode only allows the values 0 to 3 for selecting colors from the color palette.

If the graphic mode is active during character output and bit 7 of the BL register is set, an exclusive OR is performed on the character pattern and the graphic pixels behind the character pattern.

The cursor remains in the same position after character output.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 0BH	BIOS
Video: Select palette	

Selects the border and background color for graphic or text mode.

Input	AH = 0BH
	BH = 0
	BL = Border/background color

Output	No output
---------------	-----------

Remarks

In graphic mode, the color value passed defines the color of both the border and background. In text mode, the background color of each character is defined individually, so the passed color value only defines the color of the screen border.

Values for the color passed can range from 0 to 15.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 0BH, Subfunction 1**BIOS**

Video: Select color palette

Selects one of the two color palettes for the 320x200 bitmapped graphic mode.

Input

AH = 0BH

BH = 1

BL = Color palette number

Output

No output

Remarks

Two color palettes are available. They have the numbers 0 and 1 and contain the following colors:

Palette 0: Green, red, yellow

Palette 1: Cyan, magenta, white

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 0CH**BIOS**

Video: Write graphic pixel

Draws a color pixel at the specified coordinates in graphic mode.

Input

AH = 0CH

AL = Pixel color value (see below)

BH = Graphics page

CX = Screen column

DX = Screen line

Output

No output

Remarks

The pixel value color parameter depends on the current graphic mode. 640x200 bitmapped mode only permits the values 0 and 1. In the 320x200 bitmapped mode, the values 0 to 3 are permitted, which generates a certain color according to the chosen

color palette. 0 represents the selected background color; 1 represents the first color of the selected color palette; 2 represents the second color of the color palette, etc.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 0DH

BIOS

Video: Read graphic pixel

Reads the color value of a pixel at the specified coordinates in the current graphic mode.

Input

AH = 0DH

DX = Screen line

CX = Screen column

Output

AL = Pixel color value

Remarks

The pixel color value parameter depends on the current graphic mode. 640x200 bitmapped mode permits the values 0 and 1 only. In the 320x200 bitmapped mode, the values 0 to 3 are permitted, which generates a certain color according to the color palette chosen. 0 represents the selected background color; 1 represents the first color of the selected color palette; 2 represents the second color of the color palette, etc.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 0EH

BIOS

Video: Write character

Writes a write character at the current cursor position in the current screen page. The new character uses the color of the character that was previously in this position on the screen.

Input

AH = 0EH

AL = ASCII code of the character

BL = Foreground color of the character (graphic mode only)

Output

No output

Remarks

This function executes control codes (e.g., bell, carriage return) instead of reading them as ASCII codes. For example, the function sounds a beep instead of printing the bell character.

After this function displays a character, the cursor position increments so the next character appears at the next position on the screen. If the function reaches the last display position, the display scrolls up one line and output continues in the first column of the last screen line.

The foreground color parameter depends on the current graphic mode. 640x200 bitmapped mode only permits the values 0 and 1. In the 320x200 bitmapped mode, the values 0 to 3 are permitted, which generates a certain color according to the chosen

color palette. 0 represents the selected background color; 1 represents the first color of the selected color palette; 2 represents the second color of the color palette, etc. The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 0FH**BIOS**

Video: Read display mode

Reads the number of the current video mode, the number of characters per line and the number of the current screen page.

Input

AH = 0FH

Output

AL = Video mode

0: 40x25 text mode, monochrome (color card)

1: 40x25 text mode, color (color card)

2: 80x25 text mode, monochrome (mono card)

3: 80x25 text mode, color (color card)

4: 320x200 4-color graphics (color card)

5: 320x200 4-color graphics (color card)

(colors represented in monochrome)

6: 640x200 2-color graphics (color card)

7: Internal mode (mono card)

AH = Number of characters per line

BH = Current screen page number

Remarks

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 13H**BIOS (AT only)**

Video: Write character string

Displays a character string on the screen, starting at a specified screen position on a specified screen page. The characters are taken from a buffer whose address passes to the function.

Input

AH = 13H

AL = Output mode (0-3)

0: Attribute in BL, retain cursor position

1: Attribute in BL, update cursor position

2: Attribute in the buffer, retain cursor position

3: Attribute in the buffer, update cursor position

BH = Screen page number
 BL = Attribute byte of the character (modes 0 and 1 only)
 BP = Offset address of the buffer
 CX = Number of characters to be displayed
 DH = Display line
 DL = Display column
 ES = Segment address of the buffer

Output No output

Remarks

Modes 1 and 3 set the cursor position following the last character of the character string. On the next call of a BIOS function for character output, the next string of characters appears following the original character string. This does not occur in the modes 0 and 2.

In modes 0 and 1, the buffer contains only the ASCII codes of the characters to be displayed. The BL register contains the color of the character string. However, in modes 2 and 3 each character has its own attribute byte when the character is stored in the buffer. The BL register doesn't have to be loaded in this mode. Even though the character string is twice as long in these modes as the number of the characters to be displayed, the CX register requires only the number of ASCII characters in the string and not the total length of the character string.

Control codes (e.g., bell) are interpreted as control codes only, and not as characters.

When the string reaches the last position on the screen, the display scrolls upward by one line and output continues in the first column of the last screen line.

The contents of the BX, CX, DX registers and the SS, CS and DS segment registers are not affected by this function. The contents of all other registers may change, especially the SI and DI registers.

Interrupt 11h

BIOS

Determine configuration

Reads the configuration of the system as recorded during the booting process.

Input No input

Output AX = Configuration

PC and XT

Bit 0: 1 if the system has one or more disk drives
 Bit 1: Unused

- Bits 2-3: RAM available on main circuit board
 - 00: 16K
 - 01: 32K
 - 10: 48K
 - 11: 64K
- Bits 4-5: Video mode after system boot
 - 00: Unused
 - 01: 40x25, color card
 - 02: 80x25, color card
 - 03: 80x25, mono card
- Bits 6-7: Number of disk drives in the system if bit 0 is equal to 1
 - 00: 1 disk drive
 - 01: 2 disk drives
 - 10: 3 disk drives
 - 11: 4 disk drives
- Bit 8: 0 when a DMA chip is present
- Bits 9-11: Number of RS-232 cards connected
- Bit 12: 1 when system has a joystick attached
- Bit 13: Unused
- Bits 14-15: Indicates the number of printers available

AT

- Bit 0: 1 if the system has one or more disk drives
- Bit 1: 1 when a math coprocessor exists in the system
- Bit 2-3: Unused
- Bit 4-5: Video mode during system boot
 - 00: Unused
 - 01: 40x25, color card
 - 02: 80x25, color card
 - 03: 80x25, mono card
- Bits 6-7: Number of disk drives in the system if bit 0 is equal to 1
 - 00: 1 disk drive
 - 01: 2 disk drives

- 10: 3 disk drives
- 11: 4 disk drives
- Bit 8: Unused
- Bits 9-11: Number of RS-232 cards connected
- Bit 12-13: Unused
- Bits 14-15: Indicates the number of printers available

Remarks

The type of PC must be known (PC, XT or AT) to properly interpret the meanings of the individual bits of the configuration word.

The memory size indicated in bits 2 and 3 of the PC/XT configuration word refers only to the main circuit board. Interrupt 12H lets you determine the total amount of available memory.

The video mode recorded in bits 4 and 5 is the mode that was activated when the system was switched on. To determine the current video mode use function 15 of interrupt 10H. The contents of the AX register are affected by this function.

Interrupt 12h	BIOS
Determine memory size	

Input No input

Output AX = Memory size in kilobytes

Remarks

The PC and the XT can accept a maximum of 640K of RAM. The AT accepts up to 14 megabytes of RAM memory beyond the 1 megabyte limit. The memory size returned by this function ignores this extended memory. To determine the memory size beyond the 1 megabyte limit, use function 88H of interrupt 15H (available only on the AT).

The contents of the AX register are affected by this function.

Interrupt 13H, Function 00H	BIOS
Diskette: Reset	

Resets the disk controller and any connected disk drives. A reset should be executed after each disk operation during which an error occurred.

Input AH = 00H

DL = 0 or 1

Output Carry flag=0: Operation completed (AH=0)

Carry flag=1: Error (AH=error code)

Remarks

The value in the DL register is unnecessary since all the disk drives execute a reset. XT and AT models use this register to determine whether a reset should be performed on the disk drives or the hard drive.

The following error codes can occur:

01H:	Function number not permitted
02H:	Address not found
03H:	Write attempt on write protected disk
04H:	Sector not found
08H:	DMA overflow
09H:	Data transmission beyond segment border
10H:	Read error
20H:	Error in disk controller
40H:	Track not found
80H:	Time out error, unit not responding

The contents of the BX, CX, DX, SI, DI, PB registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 01H	BIOS
Diskette: Read status	

Reads the status of the disk drive since the last disk operation.

Input

AH = 01H

DL = 0 or 1

Output

Carry flag=0: Operation completed (AH=0)

Carry flag=1: Error (AH=error code)

Remarks

The value in the DL register is unnecessary, since disk drives constantly return their status. XT and AT models use this register to determine whether the status of the hard drive should be checked.

The following error codes can occur:

01H:	Function number not permitted
02H:	Address not found
03H:	Write attempt on write protected disk
04H:	Sector not found
08H:	DMA overflow
09H:	Data transmission beyond segment border
10H:	Read error

- 20H: Error in disk controller
- 40H: Track not found
- 80H: Time out error, unit not responding

The contents of the BX, CX, DX, SI, DI, PB registers and the segment registers are not affected by this function. The contents of all other registers may change.

The following error codes can occur:

- 01H: Function number not permitted
- 02H: Address not found
- 03H: Write attempt on write protected disk
- 04H: Sector not found
- 08H: DMA overflow
- 09H: Data transmission beyond segment border
- 10H: Read error
- 20H: Error in disk controller
- 40H: Track not found
- 80H: Time out error, unit not responding

The contents of the BX, CX, DX, SI, DI, PB registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 02H
BIOS

Diskette: Read disk

Reads one or more disk sectors into a buffer.

Input

- AH = 02H
- AL = Number of sectors to be read
- BX = Offset address of buffer
- CH = Track number
- CL = Sector number
- DH = Disk side number (0 or 1)
- DL = Disk drive number
- ES = Buffer segment address

Output

- Carry flag=0: Operation completed (AH=0)
- Carry flag=1: Error (AH=error code)

Remarks

The number of sectors to be read into the AL register is limited to sectors which logically follow each other on a track on one side of the disk.

The following error codes can occur:

01H:	Function number not permitted
02H:	Address not found
03H:	Write attempt on a write protected disk
04H:	Sector not found
08H:	DMA overflow
09H:	Data transmission over segment border
10H:	Read error
20H:	Error in disk controller
40H:	Track not found
80H:	Time out error, drive not responding

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all the other registers may change.

Interrupt 13H, Function 03H	BIOS
Diskette: Write to disk	

Writes one or more sectors to a disk. The data to be transmitted are taken from a buffer.

Input	AH = 03H
	AL = Number of sectors to be written
	BX = Offset address of buffer
	CH = Track number
	CL = Sector number
	DH = Disk side number (0 or 1)
	DL = Disk drive number
	ES = Buffer segment address
Output	Carry flag=0: Operation completed (AH=0)
	Carry flag=1: Error (AH=error code)

Remarks

The number of sectors that can be written in the AL register is limited to sectors which logically follow each other on a track on one side of the disk.

The following error codes can occur:

01H:	Function number not permitted
02H:	Address not found
03H:	Write attempt on a write protected disk
04H:	Sector not found
08H:	DMA overflow
09H:	Data transmission over segment border
10H:	Read error
20H:	Error in disk controller
40H:	Track not found
80H:	Time out error, drive not responding

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 04H**BIOS**

Diskette: Verify disk sectors

Compares one or more sectors on disk with the data stored in a buffer. This can be used to verify the data was properly saved to disk.

Input

AH =	04H
AL =	Number of sectors to be verified
BX =	Offset address of buffer
CH =	Track number
CL =	Sector number
DH =	Disk side number (0 or 1)
DL =	Disk drive number
ES =	Buffer segment address

Output

Carry flag=0:	Operation completed (AH=0)
Carry flag=1:	Error (AH=error code)

Remarks

The number of sectors to be verified in the AL register is limited to sectors which logically follow each other on a track on one side of the disk.

The following error codes can occur:

01H:	Function number not permitted
02H:	Address not found
03H:	Write attempt on a write protected disk
04H:	Sector not found
08H:	DMA overflow
09H:	Data transmission over segment border
10H:	Read error
20H:	Error in disk controller
40H:	Track not found
80H:	Time out error, drive not responding

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 05H**BIOS**

Diskette: Format track

Formats a complete track on one side of a disk. A buffer which contains information about the sectors to be formatted must be passed to the function.

Input

AH =	05H
AL =	Number of sectors to be formatted
BX =	Offset address of buffer
CH =	Track number
DH =	Disk side number (0 or 1)
DL =	Disk drive number
ES =	Buffer segment address

Output

Carry flag = 0:	Operation completed (AH=0)
Carry flag = 1:	Error (AH=error code)

Remarks

The number of sectors to be formatted is limited to sectors which logically follow each other on a track on one side of the disk.

The buffer passed in ES:BX contains an entry consisting of four consecutive bytes for every sector to be formatted.

- 1: Track number
- 2: Page number
- 3: Logical sector number
- 4: Number of bytes in this sector:
- 0: 128 bytes
- 1: 256 bytes
- 2: 512 bytes (PC standard)
- 3: 1,024 bytes

The logical sector number increments continuously, but may not be the same as the physical sector number.

The following error codes can occur:

- 01H: Function number not permitted
- 02H: Address not found
- 03H: Write attempt on a write protected disk
- 04H: Sector not found
- 08H: DMA overflow
- 09H: Data transmission over segment border
- 10H: Read error
- 20H: Error in disk controller
- 40H: Track not found
- 80H: Time out error, drive not responding

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all the other registers may change.

Interrupt 13H, Function 15H	BIOS (AT only)
Diskette: Determine drive type	

Senses disk change and drive type. The AT supports both the standard 320/360K drives and the 1.2 megabyte drives.

Input

AH = 15H

DL = Disk drive number (0 or 1)

Output

Carry flag = 0: Operation completed (AH=unit type)

AH = 0: Device not present

AH=1: Unit does not recognize disk change

AH=2: Unit recognizes disk change

AH=3: Hard drive (see remarks below)

Carry flag=1: Error

Remarks

The AT has a controller which selectively controls 2 disk drives and a hard drive, or one disk drive and 2 hard drives. In the latter case, the first hard drive has the number 1 and can be accessed with this function.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 16H

BIOS (AT only)

Diskette: Media change

Senses a disk change. The AT supports both the standard 320/360K drives and the 1.2 megabyte drives. This function reads any disk change that may have occurred since the last disk access.

Input

AH = 16H

DL = Disk drive number (0 or 1)

Output

AH = 0: No disk change

AH = 6: Disk changed since last disk access

Remarks

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 17H

BIOS (AT only)

Diskette: Determine disk format

Determines the format of a disk. The AT's 1.2 megabyte disk drive can read both 320/360K disks and 1.2 megabyte disks. While the BIOS can determine disk format during a read or write access, it first must be informed of the format. Function 23 must be called on the AT before you can call function 5 (format).

Input

AH = 17H

AL = Format

AL=1: 320/360K format on 320/360K drive

AL=2: 320/360K format on 1.2 megabyte drive

AL=3: 1.2 Meg format on 1.2 megabyte drive

Output

Carry flag = 0: Operation completed

Carry flag = 1: Error

Remarks

The following error codes can occur:

01H:	Function number not permitted
02H:	Address not found
03H:	Write attempt on a write protected disk
04H:	Sector not found
08H:	DMA overflow
09H:	Data transmission over segment border
10H:	Read error
20H:	Error in disk controller
40H:	Track not found
80H:	Time out error, drive not responding

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 18H	BIOS (AT and higher)
Diskette: Determine disk format	

Determining:Disk format Determines the format of a disk. This function replaces function 17H for checking disk format in AT BIOS.

Input

AH =	18H
CH =	Highest track number
CL =	Highest sector number
DL =	Drive number (0 or 1)

Output

Carry flag = 0:	Operation completed
Carry flag = 1:	Error

Remarks

This function should be called before calling function 05H (Format track) for the first time, to configure the BIOS to the proper format.

Error-Codes; see function 00H.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 00H	BIOS (XT and AT only)
Hard drive: Reset	

Resets the hard drive controller and any interfaced hard drives. A reset should be executed after every hard drive operation during which an error was reported.

Input AH = 00H
 DL = 80H or 81H

Output Carry flag = 0: Operation completed (AH=0)
 Carry flag = 1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

The value in the DL register is unnecessary since all the hard drives execute a reset. XT and AT models use this register to determine whether a reset should be performed on the disk drives or on the hard drive.

The following error codes can occur:

01H: Addressed function or unit not available
02H: Address not found
04H: Sector not found
05H: Error on controller reset
07H: Error during controller initialization
09H: DMA transmission error: Segment border exceeded
0AH: Defective sector
10H: Read error
11H: Read error corrected by ECC
20H: Controller defect
40H: Search operation failed
80H: Time out, unit not responding
AAH: Unit not ready
CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 02H	BIOS (XT and AT only)
Hard drive: Read disk	

Reads the status of the hard drive since the last hard drive operation.

Input AH = 01H
 DL = 80H or 81H

Output Carry flag = 0: Operation completed (AH=0)
 Carry flag = 1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

The value in the DL register is unnecessary since the status is consistently returned for each disk drive. XT and AT models use this register to determine whether the status of the disk drives or hard drive should be checked.

The following error codes can occur:

01H: Addressed function or unit not available
 02H: Address not found
 04H: Sector not found
 05H: Error on controller reset
 07H: Error during controller initialization
 09H: DMA transmission error: Segment border exceeded
 0AH: Defective sector
 10H: Read error
 11H: Read error corrected by ECC
 20H: Controller defect
 40H: Search operation failed
 80H: Time out, unit not responding
 AAH: Unit not ready
 CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of the other registers may change.

Interrupt 13H, Function 01H	BIOS (XT and AT only)
Hard drive: Read status	

Reads one or more hard drive sectors into a buffer.

Input	AH = 02H
	AL = Number of sectors to be read (1-128)
	BX = Offset address of buffer
	CH = Cylinder number
	CL = Sector number
	DH = Read/write head number
	DL = Hard drive number (80H or 81H)
	ES = Buffer segment address
Output	Carry flag=0: Operation completed (AH=0)
	Carry flag=1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H. Since the eight bits of the CH register can address only 256 cylinders at a time, bits 6 and 7 of the CL register (sector number) form bits 8 and 9 of the cylinder number, which enables the addressing of up to 1,023 cylinders at a time.

If several sectors are being read and the system reaches the last sector of a cylinder, reading continues at the first sector of the next cylinder of the next read/write head. If the system reaches the last read/write head, reading continues on the first sector of the following cylinder on the first read/write head.

The following error codes can occur:

01H:	Addressed function or unit not available
02H:	Address not found
04H:	Sector not found
05H:	Error on controller reset
07H:	Error during controller initialization
09H:	DMA transmission error: Segment border exceeded
0AH:	Defective sector
10H:	Read error
11H:	Read error corrected by ECC
20H:	Controller defect
40H:	Search operation failed
80H:	Time out, unit not responding
AAH:	Unit not ready
CCH:	Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 02H	BIOS (XT and AT only)
Hard drive: Write to disk	

Writes one or more sectors to the hard drive. The data to be transmitted are taken from a buffer in the calling program.

Input	AH = 03H
	AL = Number of sectors to be written (1-128)
	BX = Offset address of buffer
	CH = Cylinder number
	CL = Sector number
	DH = Read/write head number
	DL = Hard drive number (80H or 81H)
	ES = Buffer segment address

Output	Carry flag=0: Operation completed (AH=0)
	Carry flag=1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

Since the eight bits of the CH register can address only 256 cylinders at a time, bits 6 and 7 of the CL register (sector number) form bits 8 and 9 of the cylinder number, enabling the addressing of up to 1,023 cylinders at a time.

If several sectors are being written and the system reaches the last sector of a cylinder, writing continues at the first sector of the next cylinder of the next read/write head. If the system reaches the last read/write head, writing continues on the first sector of the following cylinder on the first read/write head.

The following error codes can occur:

01H:	Addressed function or unit not available
02H:	Address not found
04H:	Sector not found
05H:	Error on controller reset
07H:	Error during controller initialization
09H:	DMA transmission error: Segment border exceeded
0AH:	Defective sector
10H:	Read error
11H:	Read error corrected by ECC

20H:	Controller defect
40H:	Search operation failed
80H:	Time out, unit not responding
AAH:	Unit not ready
CCH:	Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 04H	BIOS (XT and AT only)
Hard drive: Verify disk sectors	

Verifies one or more sectors of a hard drive. Unlike the corresponding floppy disk function, the data on the hard drive are not compared with the data in memory. During data storage, four check bytes are stored for every sector; these check bytes verify the contents of a sector.

Input	AH = 04H
	AL = Number of sectors to be verified (1-128)
	BX = Offset address of buffer
	CH = Cylinder number
	CL = Sector number
	DH = Read/write head number
	DL = Hard drive number (80H or 81H)
	ES = Buffer segment address

Output	Carry flag=0: Operation completed (AH=0)
	Carry flag=1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

Since the eight bits of the CH register can only address 256 cylinders at a time, bits 6 and 7 of the CL register (sector number) form bits 8 and 9 of the cylinder number, which enables the addressing of up to 1,023 cylinders at a time.

If several sectors are being verified and the system reaches the last sector of a cylinder, verification continues at the first sector of the next cylinder of the next read/write head. If the system reaches the last read/write head, verification continues on the first sector of the following cylinder on the first read/write head.

The following error codes can occur:

01H:	Addressed function or unit not available
02H:	Address not found
04H:	Sector not found

05H:	Error on controller reset
07H:	Error during controller initialization
09H:	DMA transmission error: Segment border exceeded
0AH:	Defective sector
10H:	Read error
11H:	Read error corrected by ECC
20H:	Controller defect
40H:	Search operation failed
80H:	Time out, unit not responding
AAH:	Unit not ready
CCH:	Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 05H**BIOS (XT and AT only)**

Hard drive: Format cylinder

Formats a complete cylinder (17 sectors) of a hard drive. A buffer, which contains information about the sectors to be formatted, must be passed to the function.

Input

AH =	05H
AL =	17
BX =	Offset address of buffer
CH =	Cylinder number
CL =	1
DH =	Read/write head number
DL =	Hard drive number (80H or 81H)
ES =	Buffer segment address

Output

Carry flag=0:	Operation completed (AH=0)
Carry flag=1:	Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

Since the eight bits of the CH register can only address 256 cylinders at a time, bits 6 and 7 of the CL register (sector number) form bits 8 and 9 of the cylinder number, which enables the addressing of up to 1,023 cylinders at a time.

Since a complete cylinder is always formatted, the first sector to be formatted in the CL register is always sector 1. For the same reason the number of sectors to be formatted in the AL register is always 17, since the average hard drive operates with 17 sectors per cylinder.

The buffer, whose address is passed in ES:BX, must always be at least 512 bytes long. Only the first 34 bytes of this buffer are used for formatting the 17 sectors of a cylinder. Two succeeding bytes contain information about the corresponding physical sector. Before the function call, the first byte isn't significant. After the function call the first byte indicates whether the sector could be formatted (00H) or (80H). The second byte returns the logical sector number of the physical sector and must be placed in the buffer by calling the program before the function call.

The following error codes can occur:

01H:	Addressed function or unit not available
02H:	Address not found
04H:	Sector not found
05H:	Error on controller reset
07H:	Error during controller initialization
09H:	DMA transmission error: Segment border exceeded
0AH:	Defective sector
10H:	Read error
11H:	Read error corrected by ECC
20H:	Controller defect
40H:	Search operation failed
80H:	Time out, unit not responding
AAH:	Unit not ready
CCH:	Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 08H

BIOS (XT and AT only)

Check format

Conveys the formatting information found on the hard drive.

Input	AH = 08H
	CH = Cylinder number
	CL = Sector number
	DH = Read/write head number (0=first head)
	DL = Hard drive number

Output Carry flag=0: Operation completed (AH=0)

Carry flag=1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

Since the eight bits of the CH register can address only 256 cylinders at a time, bits 6 and 7 of the CL register (sector number) form bits 8 and 9 of the cylinder number, enabling the addressing of up to 1,023 cylinders at a time.

The total capacity of the hard drive unit in bytes can be calculated with the following formula:

Capacity = Heads * Cylinders * Sectors * 512

The following error codes can occur:

01H:	Addressed function or unit not available
02H:	Address not found
04H:	Sector not found
05H:	Error on controller reset
07H:	Error during controller initialization
09H:	DMA transmission error: Segment border exceeded
0AH:	Defective sector
10H:	Read error
11H:	Read error corrected by ECC
20H:	Controller defect
40H:	Search operation failed
80H:	Time out, unit not responding
AAH:	Unit not ready
CCH:	Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 09H

BIOS (XT and AT only)

Hard drive: Adapt to foreign drives

Interfaces other hard drives for access through BIOS functions.

Input AH = 09H

DL = Number of hard drive to be interfaced (80H or 81H)

Output Carry flag = 0: Operation completed (AH=0)

Carry flag = 1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

BIOS takes the information about the hard drive to be interfaced (number of units, read/write heads, etc.) from a table. The address of this table for the hard drive unit numbered 80H is stored in interrupt vector 41H, and the unit numbered 81H is stored in interrupt 46H.

The following error codes can occur:

- 01H: Addressed function or unit not available
- 02H: Address not found
- 04H: Sector not found
- 05H: Error on controller reset
- 07H: Error during controller initialization
- 09H: DMA transmission error: Segment border exceeded
- 0AH: Defective sector
- 10H: Read error
- 11H: Read error corrected by ECC
- 20H: Controller defect
- 40H: Search operation failed
- 80H: Time out, unit not responding
- AAH: Unit not ready
- CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 0AH

BIOS (XT and AT only)

Hard drive: Extended read

Reads one or more sectors from the hard drive into a buffer. Besides the actual 512 bytes stored in the sector, the function also reads the four check bytes (ECC).

Input AH = 0AH

AL = Number of sectors to be read (1-127)

BX = Offset address of buffer

CH = Cylinder number

CL = Sector number
DH = Read/write head number
DL = Hard drive number (80H or 81H)
ES = Buffer segment address

Output Carry flag = 0: Operation completed (AH=0)

Carry flag = 1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

Normally the controller computes the four check bytes. Here the buffer reads the information direct.

Since the eight bits of the CH register can only address 256 cylinders at a time, bits 6 and 7 of the CL register (sector number) form bits 8 and 9 of the cylinder number, enabling the addressing of up to 1,023 cylinders at a time.

If several sectors are being read and the system reaches the last sector of a cylinder, reading continues at the first sector of the next cylinder of the next read/write head. If the system reaches the last read/write head, reading continues on the first sector of the following cylinder on the first read/write head.

01H: Addressed function or unit not available
02H: Address not found
04H: Sector not found
05H: Error on controller reset
07H: Error during controller initialization
09H: DMA transmission error: Segment border exceeded
0AH: Defective sector
10H: Read error
11H: Read error corrected by ECC
20H: Controller defect
40H: Search operation failed
80H: Time out, unit not responding
AAH: Unit not ready
CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 0BH	BIOS (XT and AT only)
Hard drive: Extended write	

Writes one or more sectors to the hard drive. Besides the actual 512 bytes stored in a sector, four check bytes (ECC) stored at the end of every sector are transmitted from the buffer.

Input

AH = 0BH

AL = Number of sectors to be read (1-127)

BX = Offset address of buffer

CH = Cylinder number

CL = Sector number

DH = Read/write head number

DL = Hard drive number (80H or 81H)

ES = Buffer segment address

Output

Carry flag = 0: Operation completed (AH=0)

Carry flag = 1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

Normally the controller calculates the four check bytes. Here the system reads them direct from the buffer.

Since the eight bits of the CH register can only address 256 cylinders at a time, bits 6 and 7 of the CL register (sector number) form bits 8 and 9 of the cylinder number, enabling the addressing of up to 1,023 cylinders at a time.

If several sectors are being written and the system reaches the last sector of a cylinder, writing continues at the first sector of the next cylinder of the next read/write head. If the system reaches the last read/write head, writing continues on the first sector of the following cylinder on the first read/write head.

The following error codes can occur:

01H: Addressed function or unit not available

02H: Address not found

04H: Sector not found

05H: Error on controller reset

07H: Error during controller initialization

09H: DMA transmission error: Segment border exceeded

0AH: Defective sector

10H: Read error

11H: Read error corrected by ECC

20H: Controller defect
 40H: Search operation failed
 80H: Time out, unit not responding
 AAH: Unit not ready
 CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 0CH	BIOS (XT and above)
Hard drive: Move read/write head	

Positions a hard drive read/write head over a specific cylinder. This is useful for moving the head away from a cylinder containing data, as might be done when a computer must be transported.

Input

AH = 0CH
 DL = Hard drive number (80H or 81H)
 DH = Read/write head number
 CH = Cylinder number
 CL = Sector number

Output

Carry flag=0: Operation completed
 Carry flag=1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, and the second the number 81H.

Since the 8 bits of the CH register can address only 256 cylinders, bits 6 and 7 of the sector number (CL register) form bits 8 and 9 of the cylinder number. This permits up to 1,023 cylinders to be addressed.

The read/write head is positioned automatically for these operations, so this function is not needed for normal operations.

The following error codes can occur:

01H: Addressed function or unit not available
 02H: Address not found
 04H: Sector not found
 05H: Error on controller reset
 07H: Error during controller initialization
 09H: DMA transmission error: Segment border exceeded
 0AH: Defective sector

10H: Read error
 11H: Read error corrected by ECC
 20H: Controller defect
 40H: Search operation failed
 80H: Time out, unit not responding
 AAH: Unit not ready
 CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 0DH	BIOS (XT and AT only)
Hard drive: Reset	

Resets the hard drive controller and any interfaced hard drives. A reset should be executed after every hard drive operation during which an error was reported.

Input	AH = 0DH
	DL = Hard drive drive number (80H or 81H)
Output	Carry flag=0: Operation completed (AH=0)
	Carry flag=1: Error (AH=error code)

Remarks

The value in the DL register is unnecessary since all the hard drives execute a reset. XT and AT models use this register to determine whether a reset should be performed on the disk drives or on the hard drive.

This function is identical to function 0 listed above.

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

The following error codes can occur:

01H: Addressed function or unit not available
 02H: Address not found
 04H: Sector not found
 05H: Error on controller reset
 07H: Error during controller initialization
 09H: DMA transmission error: Segment border exceeded
 0AH: Defective sector
 20H: Controller defect
 40H: Search operation failed

80H: Time out, unit not responding

AAH: Unit not ready

CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 0EH	BIOS (PS/2)
Hard drive: Read from test buffer	

Performs a data transmission test between controller and CPU in the PS/2.

Interrupt 13H, Function 0FH	BIOS (PS/2)
Hard drive: Write to test buffer	

Performs a data transmission test between controller and CPU in the PS/2.

Interrupt 13H, Function 12H	BIOS (PS/2)
Hard drive: Controller RAM diagnostic	

Performs an internal controller diagnostic in the PS/2.

Interrupt 13H, Function 13H	BIOS (PS/2)
Hard drive: Drive diagnostic	

Performs an internal drive/controller diagnostic in the PS/2.

Interrupt 13H, Function 10H	BIOS (XT and AT only)
Hard drive: Drive ready?	

Determines if the drive is ready (i.e., the last operation has been completed and the drive can perform the next task).

Input AH = 10H

DL = Hard drive drive number (80H or 81H)

Output Carry flag = 0: Drive ready (AH=0)

Carry flag = 1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

The following error codes can occur:

01H: Addressed function or unit not available

02H: Address not found

04H: Sector not found

05H: Error on controller reset
 07H: Error during controller initialization
 09H: DMA transmission error: Segment border exceeded
 0AH: Defective sector
 10H: Read error
 11H: Read error corrected by ECC
 20H: Controller defect
 40H: Search operation failed
 80H: Time out, unit not responding
 AAH: Unit not ready
 CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 11H	BIOS (XT and AT only)
Hard drive: Recalibrate drive	

Recalibrates hard drive after an error occurs, especially after a read or write error.

Input AH = 11H
 DL = Hard drive drive number (80H or 81H)

Output Carry flag = 0: Operation completed (AH=0)
 Carry flag = 1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

The following error codes can occur:

01H: Addressed function or unit not available
 02H: Address not found
 04H: Sector not found
 05H: Error on controller reset
 07H: Error during controller initialization
 09H: DMA transmission error: Segment border exceeded
 0AH: Defective sector
 10H: Read error

11H: Read error corrected by ECC

20H: Controller defect

40H: Search operation failed

80H: Time out, unit not responding

AAH: Unit not ready

CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 14H	BIOS (XT and AT only)
Hard drive: Controller diagnostic	

Initializes an internal diagnostic test of the hard drive controller.

Input

AH = 14H

DL = Hard drive drive number (80H or 81H)

Output

Carry flag=0: Operation completed (AH=0)

Carry flag=1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H. The following error codes can occur:

01H: Addressed function or unit not available

02H: Address not found

04H: Sector not found

05H: Error on controller reset

07H: Error during controller initialization

09H: DMA transmission error: Segment border exceeded

0AH: Defective sector

10H: Read error

11H: Read error corrected by ECC

20H: Controller defect

40H: Search operation failed

80H: Time out, unit not responding

AAH: Unit not ready

CCH: Write error

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 13H, Function 15H	BIOS (AT only)
Hard drive: Determine drive type	

Determines whether the computer hardware assigned numbers 80H and 81H are hard drives. The AT contains a controller capable of controlling both hard drives and disk drives. This controller can manage either two disk drives and one hard drive, or one disk drive and two hard drives.

Input AH = 15H
 DL = Hard drive drive number (80H or 81H)

Output Carry flag = 0: Operation completed (AH=drive type)

- 0: Equipment not available
- 1: Drive does not recognize disk change
- 2: Drive recognizes disk change
- 3: Hard drive unit

 Carry flag = 1: Error (AH=error code)

Remarks

The first hard drive is assigned the number 80H, the second is assigned the number 81H.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 14H, Function 00H	BIOS
Serial port: Initialize	

Initializes and configures a serial port. This configuration includes parameters for word length, baud rate, parity and stop bits.

Input AH = 00H
 DX = Number of serial port (0=first serial port, 1=second serial port)
 AL = Configuration parameters

 Bits 0-1: Word length

- 10(b) = 7 bits
- 11(b) = 8 bits

 Bit 2: Number of stop bits

- 00(b) = 1 stop bit
- 01(b) = 2 stop bits

Bits 3-4: Parity

00(b) = none

01(b) = odd

11(b) = even

Bits 5-7: Baud rate

000(b) = 110 baud

001(b) = 150 baud

010(b) = 300 baud

011(b) = 600 baud

100(b) = 1200 baud

101(b) = 2400 baud

110(b) = 4800 baud

111(b) = 9600 baud

Output

AH = Serial port status

Bit 0: Data ready

Bit 1: Overrun error

Bit 2: Parity error

Bit 3: Framing error

Bit 4: Break discovered

Bit 5: Transmission hold register empty

Bit 6: Transmission shift register empty

Bit 7: Time out

AL = Modem status

Bit 0: Modem ready to send status change

Bit 1: Modem on status change

Bit 2: Telephone ringing status change

Bit 3: Connection to receiver status change

Bit 4: Modem ready to send

Bit 5: Modem on

Bit 6: Telephone ringing

Bit 7: Connection to receiver modem

Remarks

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all the other registers may change.

Interrupt 14H, Function 01H	BIOS
Serial port: Send character	

Sends a character to the serial port.

Input

AH = 01H

DX = Number of serial port (0=first serial port, 1=second serial port)

AL = Character code to be sent

Output

AH: Bit 7 = 0: Character transmitted

Bit 7 = 1: Error

Bit 0-6: Serial port status

Bit 0: Data ready

Bit 1: Overrun error

Bit 2: Parity error

Bit 3: Framing error

Bit 4: Break discovered

Bit 5: Transmission hold register empty

Bit 6: Transmission shift register empty

Remarks

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 14H, Function 02H	BIOS
Serial port: Read character	

Receives a character from the serial port.

Input

AH = 02H

DX = Number of serial port (0=first serial port, 1=second serial port)

Output

AH: Bit 7 = 0: Character received:

AL = Character received

Bit 7 = 1: Error:

Bit 0-6: Serial port status:

- Bit 0: Data ready
- Bit 1: Overrun error
- Bit 2: Parity error
- Bit 3: Framing error
- Bit 4: Break discovered
- Bit 5: Transmission hold register empty
- Bit 6: Transmission shift register empty

Remarks

This function should only be called if function 3 has determined that a character is ready for reception.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 14H, Function 03H**BIOS**

Serial port: Read status

Reads the status of the serial port.

Input

AH = 03H

DX = Number of the serial port (the first serial port has the number 0)

Output

AH = Serial port status

Bit 0: Data ready

Bit 1: Overrun error

Bit 2: Parity error

Bit 3: Framing error

Bit 4: Break discovered

Bit 5: Transmission hold register empty

Bit 6: Transmission shift register empty

AL = Modem status:

Bit 0: Modem ready to send status change

Bit 1: Modem on status change

Bit 2: Telephone ringing status change

Bit 3: Connection to receiver status change

Bit 4: Modem ready to send

- Bit 5: Modem on
- Bit 6: Telephone ringing
- Bit 7: Connection to receiver modem

Remarks

This function should always be called before calling function 2 (read character).

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 15H, Function 83H	BIOS (AT only)
Cassette interrupt: Set flag after time interval	

Sets bit 7 of a flag to 1 after a certain amount of time in microseconds elapses.

- Input**
- AH = 83H
 - ES = Segment address of the flag
 - BX = Offset address of the flag
 - CX = High word of elapsed time in microseconds
 - DX = Low word of elapsed time in microseconds

Output No output

Remarks

A microsecond is a millionth of a second.

The contents of the BX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 15H, Function 84H, Subfunction 0	BIOS (AT only)
Cassette interrupt: Read joystick switch settings	

Reads the status of switches on joysticks interfaced to a PC, if game ports and joysticks are present.

- Input**
- AH = 84H
 - DX = 0

Output

- Carry flag = 1: No game port connected
- Carry flag = 0: Game port present:
- AL = Switch settings:

Bit 7=1: First joystick's first switch enabled

Bit 6=1: First joystick's second switch enabled

Bit 5=1: Second joystick's first switch enabled

Bit 4=1: Second joystick's second switch enabled

Remarks

Subfunction 1 reads the joystick position(s).

The contents of the BX, CX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 15H, Function 84H, Subfunction 1	BIOS (AT only)
Cassette interrupt: Read joystick position	

Reads the positions of joysticks interfaced to a PC if game ports and joysticks are present.

Input AH = 84H

DX = 1

Output Carry flag = 1: No game port connected

Carry flag = 0: Game port present:

AX = X-position of first joystick

BX = Y-position of first joystick

CX = X-position of second joystick

DX = Y-position of second joystick

Remarks

Subfunction 0 reads the joystick switch status.

The contents of the SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 15H, Function 85H	BIOS (AT only)
Cassette interrupt: <u>Sys Req</u> key activated	

Responds to pressing or releasing of the Sys Req key. The keyboard routine calls this function.

Input AH = 85H

AL = 0: Sys Req key depressed

AL = 1: Sys Req key released

Output No output

Remarks

This function acts as an intermediary for application programs, so the application program will respond appropriately when the user presses the Sys Req key.

Interrupt 15H, Function 86H	BIOS (AT only)
Cassette interrupt: Wait	

Returns control to the calling program after a certain amount of time has elapsed.

Input

AH = 86H

CX = High word of pause time in microseconds

DX = Low word of pause time in microseconds

Output No output

Remarks

A microsecond is a millionth of a second.

The contents of the BX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 15H, Function 87H	BIOS (AT only)
Cassette interrupt: Move memory areas	

Moves areas of RAM from below the 1 megabyte limit to the range above the 1 megabyte limit, and from above the 1 megabyte limit to below the 1 megabyte limit.

Input

AH = 87H

CX = Number of words to move

ES = Segment address of global descriptor table

SI = Offset address of global descriptor table

Output

Carry flag = 0: No error

Carry flag = 1: Error:

AH=1: RAM parity error

AH=2: Incorrect GDT on function call

AH=3: Protected mode could not be initialized

Remarks

Only words can be transferred; individual bytes cannot be transferred.

Maximum amount of memory allowed in a transfer is 64K. The value in the CX register cannot exceed 8000H.

All interrupts are disabled during the memory block move.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 15H, Function 88h	BIOS (AT only)
Cassette interrupt: Determine memory size beyond 1 megabyte	

Determines the amount of memory installed beyond the 1 megabyte limit.

Input AH = 88H

Output AX = Memory size

Remarks

The value in the AX register represents memory in kilobytes (K).

Memory size below the 1 megabyte limit can be determined using interrupt 12H.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 15H, Function 89H	BIOS (AT only)
Cassette interrupt: Switch to virtual mode	

Switches the 80286 processor to virtual mode.

Input AH = 89H

Output No output

Remarks

This function should be called only if you know how virtual mode operates. Improper use of this function can easily lead to a system crash.

Interrupt 16H, Function 00H	BIOS
Keyboard: Read character	

Reads a character from the keyboard buffer. If the buffer doesn't contain a character, the function waits until a character is entered. Then the character is read and removed from the keyboard buffer.

Input AH = 00H

Output AL = 0: Extended key code

 AH = Extended key code

 AL > 1: Normal key activated

 AL = ASCII code of key

 AH = Scan code of key

Remarks

ASCII code definition occurs independent of the keyboard. Scan codes apply only to the type of keyboard attached to the PC.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 16H, Function 01H	BIOS
Keyboard: Read keyboard for character	

Reads the keyboard buffer for a character ready to be entered. If a character is available, the function passes the character to the calling function. The character remains in the keyboard buffer and can be re-read when a program calls either function 0 (see above) or function 1. The function returns to the calling program immediately after the call.

Input AH = 01H

Output Zero flag = 1: No character in the keyboard buffer
Zero flag = 0: Character available in keyboard buffer:
AL = 0: Extended key code
AH = Extended key code
AL > 1: Normal key
AL = ASCII code of the key
AH = Scan code of the key

Remarks

ASCII code definition occurs independent of the keyboard. Scan codes only apply to the type of keyboard attached to the PC.

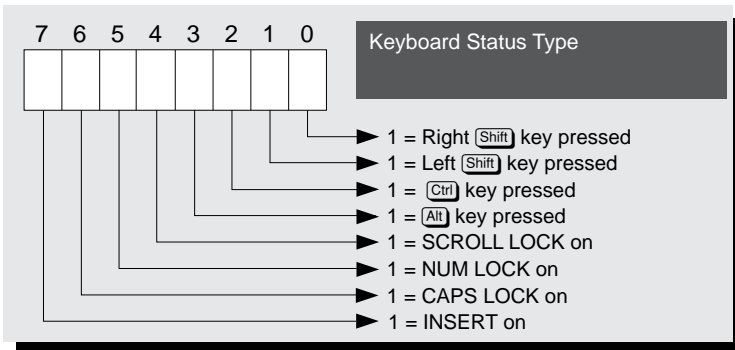
The contents of the CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 16H, Function 02H	BIOS
Keyboard: Read	

Reads and returns the status of certain control keys and various keyboard modes.

Input AH = 02H

Output AL = Keyboard status

*Remarks*

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 16H, Function 03H**BIOS (AT only)**

Keyboard: Set typematic and delay rate

Sets typematic rate and delay rate for the keyboard.

Input

AH = 03H

AL = 05H

BH = Delay before typematic

L = Typematic (repetition) rate

Output

No output

Remarks

Not every BIOS supports this function.

The following values specify the delay rate in the BH register:

Code	Delay rate
00H	1/4 second
01H	1/2 second
10H	1/4 second
11H	1 second

The following values specify the typematic rate in the BL register:

Code	RPS*	Code	RPS*	Code	RPS*	Code	RPS*
1FH	2.0	17H	4.0	0FH	8.0	07H	16.0
1EH	2.1	16H	4.3	0EH	8.6	06H	17.1
1DH	2.3	15H	4.6	0DH	9.2	05H	18.5
1CH	2.5	14H	5.0	0CH	10.0	04H	20.0
1BH	2.7	13H	5.5	0BH	10.9	03H	21.8
1AH	3.0	12H	6.0	0AH	12.0	02H	24.0
19H	3.3	11H	6.7	09H	13.3	01H	26.7
18H	3.7	10H	7.5	08H	15.0	00H	30.0
* Repetitions Per Second							

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 16H, Function 05H
BIOS (AT and above)

Keyboard: Simulate keypress

Simulates a keypress by placing the specified key code at the end of the keyboard buffer.

Input

AH = 05H

CH = Scan code of key

CL = ASCII code of key

Output

AL = 00H: O.K.

AL = 01H: Keyboard buffer full, error

Remarks

Not every BIOS supports this function.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 16H, Function 10H
BIOS (AT and above)

Keyboard: Read key on extended keyboard

Reads the keyboard buffer. This function is similar to function 00H, except that function 10H is intended for extended keyboards with 101 or 102 keys (i.e., keyboards including the **F11** and **F12** keys).

Input

AH = 10H

Output

AH = Scan code of key

AL = ASCII code of key

Remarks

Not every BIOS supports this function.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 16H, Function 11H	BIOS (since AT)
Keyboard: Read extended keyboard for character	

Reads the keyboard buffer for a character ready to be entered. This function is similar to function 01H, except that function 11H is intended for extended keyboards with 101 or 102 keys (i.e., keyboards including the **F11** and **F12** keys).

Input	AH = 11H
Output	Zero flag=1: No character in the keyboard buffer
	Zero flag=0: Character available
	AL = 0: Extended key code
	AH=Extended key code
	AL > 0: Normal key activated
	AL=ASCII code of key
	AH=Scan code of key

Remarks

Not every BIOS supports this function.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 17H, Function 00H	BIOS
Keyboard: Write character	

Writes a character to one of the printers interfaced to the PC.

Input	AH = 00H
	AL = Character code to be printed
	DX = Printer number
Output	AH = Printer status:
	Bit 0=1: Time out error
	Bit 1: Unused
	Bit 2: Unused
	Bit 3=1: Transfer error

Bit 4=0: Printer offline

Bit 4=1: Printer online

Bit 5=1: Printer out of paper

Bit 6=1: Receive mode selected

Bit 7=0: Printer busy

Remarks

Parallel port LPT1 is assigned the number 0, parallel port LPT2 is assigned the number 1 and parallel port LPT3 is assigned the number 2.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 17H, Function 01H

BIOS

Printer: Initialize printer

Printer: Initialize printer

Initializes the printer interfaced to the PC. This function should be executed before executing function 0 (see above).

Input

AH = 01H

DX = Printer number

Output

AH = Printer status

Remarks

Parallel port LPT1 is assigned the number 0, parallel port LPT2 is assigned the number 1 and parallel port LPT3 is assigned the number 2.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 17H, Function 02H

BIOS

Printer: Read printer status

Returns the status of the printer interfaced to the PC.

Input

AH = 02H

DX = Printer number

Output

AH = Printer status

Remarks

Parallel port LPT1 is assigned the number 0, parallel port LPT2 is assigned the number 1 and parallel port LPT3 is assigned the number 2. The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 18H	BIOS
Call ROM BASIC	

Accesses BASIC in ROM if a system disk cannot be found during the system bootstrap process.

Input No input

Output No output

Remarks

Very few PCs or compatibles have built-in ROM BASIC (this is a throwback from the early days of the PC). If a PC doesn't have ROM BASIC, interrupt 18H returns the system to the calling program. However, if the PC does have ROM BASIC, interrupt 18H calls BASIC. In most cases, the only way to return to DOS is by warm-starting the computer (pressing the **Ctrl** **Alt** **Del** keys) or turning the computer off and on again. Some versions of ROM BASIC allow an exit to DOS by entering the SYSTEM command from BASIC.

Interrupt 19H	BIOS
Boot process	

Boots the computer.

Input No input

Output No output

Remarks

Pressing the **Ctrl** **Alt** **Del** keys invokes this interrupt from the keyboard.

Interrupt 1AH, Function 00H	BIOS
Date and time: Read clock count	

Reads the current clock count. The clock count increments 18.2 times per second. This calculates the time elapsed since the computer was switched on.

Input AH = 00H

Output CX = High word of the clock count

DX = Low word of the clock count

AL = 0: Less than 24 hours have elapsed since the last reading

AL > 0: More than 24 hours have elapsed since the last reading

Remarks

The AT, which has a battery powered realtime clock, sets the clock count to the current time when the computer boots. PCs (which don't have realtime clocks) set the counter to 0 during booting.

The contents of the BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 1AH, Function 01H	BIOS
Date and time: Set clock count	

Sets the contents of the current clock count, which increments 18.2 times per second. This calculates the time elapsed since the computer was switched on and sets the current time through this function.

Input AH = 01H

CX = High word of clock count

DX = Low word of clock count

Output No output

Remarks

The AT, which has a battery powered realtime clock, sets the clock count to the current time when the computer boots. PCs (which don't have realtime clocks) set the counter to 0 during booting. PC owners should use this function to set the current time.

The contents of the AX, BX, CX, DX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 1AH, Function 02H	BIOS (AT only)
Date and time: Read realtime clock	

Reads the time from the realtime clock.

Input AH = 02H

Output Carry flag = 0: O.K.:

CH = Hours

CL = Minutes

DH = Seconds

Carry flag = 1: Dead clock battery

Remarks

All time readings appear in BCD format.

The contents of the BX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 1AH, Function 03H	BIOS (AT only)
Date and time: Set realtime clock	

Sets the time on the realtime clock.

Input

AH = 03H

CH = Hours

CL = Minutes

DH = Seconds

DL = 1: Daylight Saving Time

DL = 0: Standard Time

Output No output

Remarks

All time settings must be in BCD format.

The contents of the BX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 1AH, Function 04H

BIOS (AT only)

Date and time: Read date from realtime clock

Reads the current date from the realtime clock.

Input AH = 04H

Output Carry flag = 0: O.K.:

CH = Century (19 or 20)

CL = Year

DH = Month

DL = Day

Carry flag = 1: Dead clock battery

Remarks

All date readings appear in BCD format.

The contents of the BX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 1AH, Function 05H

BIOS (AT only)

Date and time: Set date in realtime clock

Sets the current date in the realtime clock.

Input AH = 05H

CH = Century (19 or 20)

CL = Year
DH = Month
DL = Day

Output No output

Remarks

All date settings must be in BCD format.

The contents of the BX, CX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 1AH, Function 06H

BIOS (AT only)

Date and time: Set alarm time

Sets alarm time for the current day. The alarm time triggers interrupt 4AH.

Input AH = 06H
CH = Hours
CL = Minutes
DH = Seconds

Output Carry flag=0: O.K.
Carry flag=1: Dead clock battery or programmed alarm time

Remarks

All alarm settings must be in BCD format.

During booting, interrupt 4AH points to an IRET command. If this interrupt doesn't point to a particular routine responding to the alarm, nothing will happen once the alarm time is reached.

Only one alarm time can be active at a time. If another alarm setting already exists, you must first delete it by using interrupt 26-1AH, function 7 (see below).

The contents of the BX, CX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 1AH, Function 07H

BIOS (AT only)

Date and time: Reset alarm time

Clears an existing alarm setting created by using function 06H above.

Input AH = 07H

Output No output

Remarks

This function must be called when you want to change an alarm setting. Reset the alarm, then use function 06H to set the new alarm time.

The contents of the BX, CX, SI, DI, BP registers and the segment registers are not affected by this function. The contents of all other registers may change.

Interrupt 1BH**BIOS/DOS**Keyboard: **Break** key pressed

Records the occurrence of a **Ctrl Break** key combination and triggers interrupt 1BH. During the system boot, BIOS sets interrupt 1BH to an IRET command to prevent any reaction.

This routine sets a flag to indicate the user has pressed **Ctrl Break**. Following the execution of one of the DOS functions, this flag is tested for character input or output. If the system encounters **Ctrl Break**, the current program stops. In addition, when a batch file is in process, the program asks whether the batch file should be continued or terminated.

Pressing **Ctrl C** doesn't activate the interrupt. This key combination forces DOS to end the currently executing program. However, the DOS functions for character input/output search for this key combination.

To prevent termination of an application program, this interrupt can also be pointed to a user routine by pressing **Break** or **Ctrl Break**.

Input No input**Output** No output*Remarks*

Before returning control to the calling program, this interrupt must restore all registers to their previous values.

Interrupt 1CH**BIOS**

Periodic interrupt

The timer IC calls interrupt 8H approximately 18.2 times per second. After ending its task, it calls interrupt 1CH to allow an application program access to the signals from the timer IC. During booting, BIOS initializes the interrupt vector of interrupt 1CH so it points to an IRET command, which prevents any response if the interrupt is called. For example, this interrupt can be pointed to a user routine to create a constant display clock on the screen.

Input No input**Output** No output*Remarks*

This interrupt must restore all registers to their previous values before returning control to the calling program.

Interrupt 1DH**BIOS**

Video table

Sets a pointer to a table. The vector of this interrupt in the vector table, starting at address 0000:0074, stores the offset and segment address of this table. The table itself contains a collection of parameters used by BIOS for initializing a certain video

mode. This involves the 16 memory locations on the video card, whose heart is a 6845 video processor. For this reason the table to which the vector points and which is part of the ROM-BIOS, consists of 16 consecutive bytes that indicate the contents of individual registers for a certain video mode. The first of these 16 bytes is copied into the first register of the 6845, the second byte into the second register, etc. The table in ROM contains a total of four 16-byte entries: 40x25 color mode, 80x25 color mode, 80x25 monochrome mode and one entry for the various color graphics modes.

Do not call this interrupt. If you do, the system will attempt to read the video table as executable code and will crash.

Input No input

Output No output

Interrupt 1EH	BIOS/DOS
Drive table	

Sets a pointer to a table. The vector of this interrupt in the vector table starting at address 0000:0078 stores the offset and segment address of this table. The table itself contains a collection of parameters used by BIOS in disk drive access. BIOS has a table in ROM, but deviates the interrupt vector of interrupt 30 to its own table which allows faster disk access than the BIOS table.

Do not call this interrupt. If you do call it, the system will attempt to read the drive table as executable code and will crash.

Input No input

Output No output

Interrupt 1FH	BIOS/DOS
Character table	

Sets a pointer to a table. The vector of this interrupt in the vector table, starting at address 0000:007C, stores the offset and segment address of this table. The table itself contains character patterns for the characters possessing ASCII codes 128 to 255. BIOS needs this table to display the graphic mode characters on the screen. These characters are displayed by placing the character patterns, which are stored in this table, on the screen as individual pixels.

Since the character patterns for codes 0 to 127 are already stored in a table in ROM-BIOS, this table contains only the character patterns for codes 128 to 255. The DOS GRAFTABL command loads a table for codes 127 to 255 into RAM and points the interrupt vector of interrupt 31 to this table. A user table can be added to display on the screen, in graphic mode, certain characters that are not part of the normal PC character set. The construction of the table requires that eight consecutive bytes define the appearance of the character. The first eight bytes of the table define the appearance of ASCII code 128, the next eight bytes define ASCII code 129, etc. Each set of eight bytes represents the eight lines which denote a character in graphic mode. The eight bits of each byte indicate the eight columns of pixels for each line.

Do not call this interrupt. If you do call it, the system will attempt to read the character table as executable code and will crash.

Input No input

Output No output

EGA/VGA BIOS Functions

EGA and VGA cards enhance the BIOS Video-Interrupt 10H with numerous functions, which provide access to the extended capabilities of the cards. A description of these functions, which are called like the normal BIOS-functions of the Interrupt 10H, can be found in Appendix B.

Interrupt 10H, Function 00H	EGA/VGA
Screen: Set video mode	

Sets and initializes the video mode.

Input

AH = 00H

AL = EGA/VGA video mode

AL=0: 40x25-character text, 16 colors (EGA/VGA - color monitor)

AL=1: 40x25-character text, 16 colors (EGA/VGA - color monitor)

AL=2: 80x25-character text, 16 colors (EGA/VGA - color monitor)

AL=3: 80x25-character text, 16 colors (EGA/VGA - color monitor)

AL=4: 320x200 graphics, 4 colors (EGA/VGA - color monitor)

AL=5: 320x200 graphics, 4 colors (EGA/VGA - color monitor)

AL=6: 640x200 graphics, 2 colors (EGA/VGA - color monitor)

AL=7: 80x25-character text, mono (EGA/VGA - mono monitor)

AL=13: 320x200 graphics, 16 colors (EGA/VGA - color monitor)

AL=14: 640x200 graphics, 16 colors (EGA/VGA - color monitor)

AL=15: 640x350 graphics, mono (EGA/VGA - mono monitor)

AL=16: 640x350 graphics, 4 colors (64K EGA - high-res monitor)

640x350 graphics, 16 colors (128K EGA/VGA - high-res monitor)

AL=17: 640x480 graphics, 2 colors (VGA only)

AL=18: 640x480 graphics, 16 colors (VGA only)

AL=19: 320x200 graphics, 256 colors (VGA only)

Output

No output

Remarks

Modes 0 and 1, 2 and 3, 4 and 5 differ in the output of the color signal that is suppressed in the first mode. This isn't possible on an EGA/VGA card so the modes are identical. If bit 7 of the AL register is set when this function is called, the contents of the video RAM will not be erased when the new mode is enabled. The task is to program the video controller and define a color palette. The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 01H	EGA/VGA
Screen: Define cursor appearance	

Defines the starting and ending lines of the screen cursor. This function is independent of the .i.display page; being displayed.

Input	AH = 01H
	CH = Starting line of the cursor
	CL = Ending line of the cursor

Output	No output
---------------	-----------

Remarks:

Since the possible values depend on the size of the current video mode's character matrix, the values in the CH and CL registers always refer to an eight-line character matrix. The values should thus be between zero and seven. The EGA/VGA-BIOS adapts these values to the current size of its own character matrix.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 02H	EGA/VGA
Screen: Position cursor	

Moves the cursor into position on the screen.

Input	AH = 02H
	BH = Video page number
	DH = Screen line
	DL = Screen column

Output	No output
---------------	-----------

Remarks:

The cursor moves only if the specified display page is the current page.

The values for the screen line and column are based on the resolution of the current display mode.

Assigning the DH and DL registers values for a non-existent screen position (e.g., column 0, line 255) makes the cursor disappear from the screen.

The number of the display page is based on how many display pages the card has available.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 03H	EGA/VGA
Screen: Read cursor position	

Reads the position of the text cursor on the screen and the starting and ending lines of the screen cursor.

Input	AH = 03H
	BH = Video page number
Output	DH = Screen line in which cursor is located
	DL = Screen column in which cursor is located
	CH = Starting line of screen cursor
	CL = Ending line of screen cursor

Remarks:

The screen line and screen column parameters refer to the text coordinate system, even if a graphic mode is active.

The starting and ending lines of the cursor are returned correctly only in the text modes. They have no meanings in graphic modes.

The contents of registers BX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 04H	EGA/VGA
Screen: Read lightpen position	

Reads the lightpen's position on the screen. This is only possible on EGA cards, since VGA cards do not support lightpens.

Input	AH = 04H
Output	AH = 0: Lightpen position cannot be read
	AH = 1: Lightpen position read
	DH = Screen row (text mode)
	DL = Screen column (text mode)
	CH = Screen row (graphic mode)
	BX = Screen column (graphic mode)

Remarks:

This function call must be repeated until a 1 is returned in the AH register, since only then can the lightpen position be determined.

Graphic mode returns lightpen coordinates less accurately than in text mode. The Y-coordinate (row) is always a multiple of 2. For example, function 04H cannot differentiate between row 8 or row 9. In the 320x200 pixel graphic mode, the X-coordinate (column) is always a multiple of 4, and in the 640x200 pixel graphic mode the X-coordinate is a multiple of 8.

The contents of the CL register and the contents of the segment registers SS, CS and DS are not affected by this function. The content of all other registers may change, especially the SI and DI registers.

Interrupt 10H, Function 05H	EGA/VGA
Screen: Select current display page	

Selects the current display page, and thereby the page which appears on the screen (text mode only).

Input AH = 05H
 AL = Display page number

Output No output

Remarks:

The number of available display pages depends on the amount of video RAM installed on the EGA/VGA card.

When a new page is selected the screen cursor will be moved to the position of the text cursor on this page.

Switching between different pages does not change the contents of these pages.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 06H	EGA/VGA
Screen: Scroll text lines up	

Scrolls part of the current display page up by one or more lines.

Input AH = 06H
 AL = Number of lines to be scrolled up
 AL=0: Clear window
 CH = Screen line of upper left corner of window
 CL = Screen column of upper left corner of window
 DH = Screen line of lower right corner of window
 DL = Screen column of lower right corner of window
 BH = Color (attribute) for blank line(s)

Output No output

Remarks:

Normally the contents of the current display page are scrolled, but in the 320x200 four-color graphic mode this function only affects display page 0.

Clearing the screen window (number of lines = 0) is the same as filling it with spaces (ASCII code 32).

The contents of the lines scrolled out of the window are lost and cannot be recovered.

Use function 0 of this interrupt to clear the screen.

Appendix B: EGA/VGA BIOS Functions

B - 5

The interpretation of the attribute byte in the BL register depends on the current video mode. In text mode it is interpreted as any other attribute byte in video RAM. In 640x200 two-color mode this byte represents the color value for eight successive pixels. In 320x200 four-color mode this byte represents the color value of four successive pixels. In all other graphic modes it represents the color of all the pixels in the cleared screen area.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 07H

EGA/VGA

Screen: Scroll text lines down

Scrolls part of the current display page down one or more lines.

Input

AH = 07H

AL = Number of lines to be scrolled down

AL=0: Clear window

CH = Screen line of upper left corner of window

CL = Screen column of upper left corner of window

DH = Screen line of lower right corner of window

DL = Screen column of lower right corner of window

BH = Color (attribute) for blank line(s)

Output

No output

Remarks:

Normally the contents of the current display page are scrolled, but in 320x200 four-color graphic mode this function only affects display page 0.

Clearing the screen window (number of lines = 0) is the same as filling it with spaces (ASCII code 32).

The contents of the lines scrolled out of the window are lost and cannot be recovered.

To clear the entire screen, use function 0 of this interrupt instead.

The interpretation of the attribute byte in the BL register depends on the current display mode. In the text mode it is interpreted like any other attribute byte in the video RAM. In the 640x200 two-color mode this byte represents the color value for eight successive pixels. In the 320x200 four-color mode it represents the color value of four successive pixels. In all other graphic modes it represents the color of all the pixels in the cleared screen area.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 08H

EGA/VGA

Screen: Read character/color

Reads and returns the ASCII code and color (attribute) of the character at the current cursor position.

Input

AH = 08H

BH = Video page number

Output AL = ASCII code of character

AH = Color (attribute)

Remarks:

This function can also be called in the graphic mode, whereby the bit pattern of the character on the screen will be compared with the bit patterns of the characters. If the character cannot be identified, the AL register will contain the value zero after the call.

In the 320x200 four-color graphic mode this function only affects display page 0.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 09H	EGA/VGA
Screen: Write character/color	

Writes character with the specified color at the current cursor position (in a specified display page).

Input AH = 09H
 BH = Video page number
 CX = Repeat factor
 AL = ASCII code of character
 BL = Attribute

Output No output

Remarks:

If the graphic mode is active and the specified character is to be printed more than once (the value of the CX register is greater than 1), all the characters must fit on the current screen line.

In the 320x200 four-color graphic mode this function correctly works only on display page 0.

Within a graphic mode the attribute in the BL register specifies the foreground color of the character, whereby the background color is zero. If bit seven is set, the character will be XORed with the bitmap at the output position.

The controls codes for bell, carriage return, etc. are not recognized as control codes, and are displayed as normal ASCII characters.

This function can also be used to output characters in the graphic mode, in which case the character patterns are taken from one of the EGA character tables.

This function does not move the cursor to the next screen position.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 0AH	EGA/VGA
Screen: Write character	

Writes a character to the current screen position on the specified display page and the color of the old character at this position will be retained.

Input	AH = 0AH
	AL = ASCII code of the character
	BH = Video page number
	BL = Foreground color of character for graphic modes
	CX = Repeat factor

Output No output

Remarks:

If the graphic mode is active and the specified character is to be printed more than once (the value of the CX register is greater than 1), all the characters must fit on the current screen line.

The controls codes for bell, carriage return, etc. are not recognized as such and are displayed as normal ASCII characters.

This function can also be used to output characters in the graphic mode, in which case the character patterns are taken from one of the EGA character tables.

Within a graphic mode the attribute in the BL register specifies the foreground color of the character, whereby the background color is zero. If bit seven is set, the character will be XORed with the bitmap at the output position.

This function does not move the cursor to the next screen position.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 0BH, Subfunction 0

EGA/VGA

Screen: Select border/background color

Selects the border and background color for the graphic or text mode.

Input	AH = 0BH
	BH = 0
	BL = Border/background color

Output No output

Remarks:

This function should be called only when the EGA/VGA card is in the 320x200 or 640x200 graphic mode. Use function 10H for all other modes.

Bits zero to three of the BL register set the background and border color. Setting bit four will enable high-intensity colors.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 0BH, Subfunction 1

EGA/VGA

Screen: Select color palette

Selects one of the two color palettes for the 320x200 graphic mode.

Input AH = 0BH
 BH = 1
 BL = Color palette number

Output No output

Remarks:

This function should be called only when the EGA/VGA card is in the 320x200 or 640x200 graphic mode. Use function 10H for all other modes.

The EGA/VGA-BIOS emulates the two CGA color palettes with the numbers 0 and 1. They contain the following colors:

Palette 0: green, red, yellow

Palette 1: cyan, magenta, white

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 0CH	EGA/VGA
Screen: Write pixel	

Sets the color value of a screen pixel in the graphic mode.

Input AH = 0CH
 BH = Video page
 DX = Screen line
 CX = Screen column
 AL = Color value

Output No output

Remarks:

The color value depends on the colors available in the current display mode.

If bit seven of the AL register is set, the color value will be XORed with the previous color value of the pixel.

The display page is ignored in the 320x200 four-color graphic mode.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 0DH	EGA/VGA
Screen: Read pixel	

The color value of a pixel in the graphic mode is returned.

Input AH = 0DH
 BH = Video page

DX = Screen line

CX = Screen column

Output AL = Color value

Remarks:

The color value depends on the colors available in the current display mode.

The display page is ignored in the 320x200 four-color graphic mode.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 0EH

EGA/VGA

Screen: Write character

Writes a character to the current cursor position on the current display page. The color of the old character at this position will be retained.

Input AH = 0EH

AL = ASCII character code

BL = Foreground color of character

Output No output

Remarks:

This function does not treat the various control codes like bell and carriage as normal characters, and implements them as the control characters they represent.

After displaying a character with this function, the cursor position is incremented so the next character will be printed at the following screen position. If the last screen position has been reached, the screen will be scrolled up one line and the output will continue in the first column of the last screen line.

If bit seven of the BL register is set, the color value will be XORed with the previous color value of the pixels. The background color is zero.

Characters can be displayed in the graphic mode with this function. The character patterns are taken from one of the EGA character tables.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 0FH

EGA/VGA

Screen: Returns current display mode

Reads the number of the current display mode, the number of characters per line, and the number of the current display page.

Input AH = 0FH

Output	AL = Video mode:
	AL=0: 40x25-character text, 16 colors (EGA/VGA - color monitor)
	AL=1: 40x25-character text, 16 colors (EGA/VGA - color monitor)
	AL=2: 80x25-character text, 16 colors (EGA/VGA - color monitor)
	AL=3: 80x25-character text, 16 colors (EGA/VGA - color monitor)
	AL=4: 320x200 graphics, 4 colors (EGA/VGA - color monitor)
	AL=5: 320x200 graphics, 4 colors (EGA/VGA - color monitor)
	AL=6: 640x200 graphics, 2 colors (EGA/VGA - color monitor)
	AL=7: 80x25-character text, mono (EGA/VGA - mono monitor)
	AL=13: 320x200 graphics, 16 colors (EGA/VGA - color monitor)
	AL=14: 640x200 graphics, 16 colors (EGA/VGA - color monitor)
	AL=15: 640x350 graphics, mono (EGA/VGA - mono monitor)
	AL=16: 640x350 graphics, 4 colors (64K EGA - high-res monitor)
	640x350 graphics, 16 colors (128K EGA/VGA - high-res monitor)
	AL=17: 640x480 graphics, 2 colors (VGA only)
	AL=18: 640x480 graphics, 16 colors (VGA only)
	AL=19: 320x200 graphics, 256 colors (VGA only)
	AH = Number of characters per line
	BH = Number of current display page

Remarks:

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 00H	EGA/VGA
Screen: Set palette registers	

Sets the contents of a palette register in the attribute controller of the EGA/VGA card.

Input	AH = 10H
	AL = 00H
	BL = Color value
	BH = Register to be addressed
Output	No output

Remarks:

Since the register number is not checked by the BIOS, you can also program the other registers in the attribute controller. These include the mode control register, overscan register and others.

The contents of registers BX, CX, DX, SI, DI, BP, and the segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 01H	EGA/VGA
Screen: Set screen border color	

Copies resulting value into the .i.overscan register; of the .i.EGA attribute controller;.

Input	AH = 10H
	AL = 01H
	BH = Border color

Output	No output
---------------	-----------

Remarks:

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 02H	EGA/VGA
Screen: Set all palette registers	

Configures all 16 palette registers and the .i.overscan register;.

Input	AH = 10H
	AL = 02H
	ES = Segment address of color table
	DX = Offset address of color table

Output	No output
---------------	-----------

Remarks:

The ES:BX register pair points to a 17-byte table. The first 16 bytes will be transferred to the 16 palette registers of the attribute controller and the 17th byte will be copied into the overscan register.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 03H	EGA/VGA
Screen: Set blinking attribute	

Determines whether bit 7 in the attribute byte of a character in the text mode will enable character blinking, or display characters on a high-intensity background.

Input AH = 10H
 AL = 00H
 BL = Blinking attribute
 BL=0: high-intensity background
 BL=1: blinking

Output No output

Remarks:

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 07H	VGA
Screen: Read out palette register	

Reads the contents of one of the attribute controller's palette registers.

Input AH = 10H
 AL = 07H
 BH = Number of palette register

Output BL = Contents of addressed palette register

Remarks:

Since the BIOS doesn't verify the number of the palette register read, this function can read all the registers of the attribute controller.

The contents of registers BL, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 08H	VGA
Screen: Read contents of overscan register	

Returns the contents of the overscan register containing the screen's border color.

Input AH = 10H
 AL = 08H

Output BH = Contents of the overscan register

Remarks:

The contents of registers BL, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 09H**VGA**

Screen: Read contents of all palette registers and overscan register

Copies the contents of the 16 palette registers and overscan register into a buffer.

Input

AH = 10H

AL = 09H

ES = Segment address of the buffer

DX = Offset address of the buffer

Output

No output

Remarks:

The buffer must be a minimum of 17 bytes long to allow room for all the palette registers (bytes 0-15) plus the overscan register (byte 16).

The contents of registers BL, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 10H**VGA**

Screen: Define a DAC color register

Allows the definition of one of the 256 available DAC color registers.

Input

AH = 10H

BX = Number of the DAC color register (0-255)

CH = Green value

CL = Blue value

DH = Red value

Output

No output

Remarks:

Only bits 0 to 5 in the CH, CL and DH registers are of importance to this function. All other bits are ignored.

The contents of registers BL, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 12H**VGA**

Screen: Load multiple DAC color registers

Allows the definition of multiple DAC color registers.

Input

AH = 10H

AL = 12H

BX = Number of the first DAC color register (0-255)

CX = Number of registers to be loaded

ES = Segment address of the buffer

DX = Offset address of the buffer

Output No output

Remarks:

The assigned buffer must be able to hold a group of three consecutive bytes for each DAC color register. The first byte contains the red value; the second byte contains the green value; and the third byte contains the blue value. These first three bytes correspond to the first DAC color register being accessed, the next three for the bytes to the next DAC color register.

Only bits 0 to 5 in the CH, CL and DH registers are of importance to this function. All other bits are ignored.

If the sum of BX and CX is greater than 255, the first DAC color register is reloaded after the last register is loaded.

The contents of registers BL, CX, DX, SI, DI, BP and all segment registers are unchanged by this function.

Interrupt 10H, Function 10H, Subfunction 13H

VGA

Screen: Select color register or select a DAC register group

Manipulates bit 7 of the mode control registers.

Input AH = 10H

AL = 13H

BL = 00H or 01H (see below)

BH = see below

Output No output

Remarks:

This subfunction performs as two different subfunctions, depending on the value contained in the BL register. Subfunction 00H allows color selection, while subfunction 01H allows the selection of the active DAC register group.

Subfunction 00H copies bit 0 in the BH register into bit 7 of the mode control register, thus providing a method of color selection. If bit 0 in the BH register contains a value of 0, then the 256 DAC color registers are divided into four groups of 64 registers. Color selection involves bits 0-5 in the corresponding palette register, as well as bits 2-3 of the color select register. These eight bits act as the index for the DAC color register. If bit 0 in the BH register contains a 1, the DAC color registers are divided into 16 groups of 16 registers. Then color selection involves the lowest 4 bits of the palette register and the lowest 4 bits of the color select register, acting as the 8-bit index to the DAC color table.

Subfunction 01H loads the color select register, whose contents are specified by the active group of DAC color registers. The contents of the BH register are copied to the color select register.

The contents of registers BL, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 15H	VGA
Screen: Read a DAC color register	

Returns the contents of one of the 256 DAC color registers.

Input	AH = 10H
	AL = 15H
	BX = Number of the DAC color registers

Output	CH = Green value
	CL = Blue value
	DH = Red value

Remarks:

Only bits 0 to 5 in the CH, CL and DH registers are of importance to this function. All other bits are ignored.

The contents of registers BX, DL, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 17H	VGA
Screen: Load contents of multiple DAC color registers	

Loads several DAC color registers at a time.

Input	AH = 10H
	AL = 17H
	BX = Number of the first DAC color register to be loaded (0-255)
	CX = Number of registers to be loaded
	ES = Segment address of buffer
	DX = Offset address of buffer

Output	No output
---------------	-----------

Remarks:

The contents of each DAC color register are represented within a buffer by three consecutive bytes. The red value is loaded into the first of these registers; the green value is loaded into the second of these registers; and the blue value is loaded into the third register. The first group of three bytes corresponds to the first DAC color register addressed, the second group to the next DAC color register, etc.

Only bits 0 to 5 in the CH, CL and DH registers are of importance to this function. All other bits are ignored.

If the sum of BX and CX is greater than 255, the first DAC color register is reloaded after the last register is loaded (wrap-around occurs).

The contents of registers BX, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 18H	VGA
Screen: Load DAC mask register	

Loads the specified value into the DAC mask register.

Input	AH = 10H
	AL = 18H
	BL = Value of DAC mask register

Output	No output
---------------	-----------

Remarks:

The contents of the DAC mask register play an important role in color selection. An AND instruction adds it to the index access to the DAC color table.

The contents of registers BH, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 19H	VGA
Screen: read out contents of the DAC mask register	

Reads the current contents of the DAC mask register.

Input	AH = 10H
	AL = 19H
Output	BL = Contents of the DAC mask register

Remarks:

The contents of the DAC mask register play an important role in color selection. An AND instruction adds it to the index access to the DAC color table.

The contents of registers BH, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 1AH	VGA
Screen: Returns method of color selection and color select register	

Returns the method of color selection, contained in the contents of bit 7 of the mode control register. It also returns the contents of the color select register chosen by the active group of DAC color registers.

Input	AH = 10H
	AL = 1AH
Output	BL = Bit 7 of mode control register
	BH = Contents of color select registers

Remarks:

The contents of registers BX, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 10H, Subfunction 1BH	VGA
Screen: Convert DAC color register into gray scales	

Converts a specified range within a DAC color table into gray scales.

Input	AH = 10H
	AL = 1BH
	BX = Number of first DAC color register to be converted
	CX = Total number of DAC color registers to be converted

Output	No output
---------------	-----------

Remarks:

Conversion into grays results from changes to the red, green and blue values, as well as the intensity of these values. The default factor for red is 0.3, the default factor for green is 0.59, and the default for blue 0.11.

The contents of registers BX, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 00H	EGA/VGA
Screen: Load user-defined character set	

Loads a user-defined character set from RAM into one of the two EGA character tables.

Input	AH = 11H
	AL = 00H
	BH = Lines per character (also bytes per character)
	BL = Character table (0 or 1)
	CX = Number of characters in table
	DX = ASCII code of first character in table
	ES = Segment address of character table in RAM
	BP = Offset address of character table in RAM

Output	No output
---------------	-----------

Remarks:

A maximum of 512 characters can be loaded per character table.

The loaded character set is not activated, nor are the .i.CRTC; registers programmed to the size of the characters. The changes will not be visible on the screen unless the character definitions are loaded into the active character table.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 01H	EGA/VGA
Screen: Load 8x14 character set	

Loads the entire 8x14-pixel character set from EGA/VGA ROM-BIOS into one of the two character set tables.

Input

AH = 11H

AL = 01H

BL = Character table (0 or 1)

Output No output

Remarks:

The loaded character set is not activated, nor are the CRTC registers programmed to the size of the characters. The changes will not be visible on the screen unless the character definitions are loaded into the active character table.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 02H	EGA/VGA
Screen: Load 8x8 character set	

Loads the entire 8x8-pixel character set from EGA/VGA ROM-BIOS into one of the two character set tables.

Input

AH = 11H

AL = 02H

BL = Character table (0 or 1)

Output No output

Remarks:

The loaded character set is not activated, nor are the CRTC registers programmed to the size of the characters. The changes will not be visible on the screen unless the character definitions are loaded into the active character table. The EGA card displays 43 lines on the screen, while the VGA card displays 50 lines.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 03H	EGA/VGA
Screen: Activate character set	

Activates one (or two) of the four 256-character character sets.

Input

AH = 11H

AL = 03H

BL = Number of the character set to activate

Output No output

Remarks:

Bits zero and one of the BL register specify the number of the character set to be accessed when bit three of the attribute byte of the character is zero.

Bits two and three of the BL register specify the number of the character set to be accessed when bit three of the attribute byte of the character is one.

If the contents of bits zero and one are identical to the contents of bits two and three of the BL register, then bit three of the character attribute byte has no effect on the character displayed. Only 256 different characters can then be displayed on the screen.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 04H

VGA

Screen: Load 8x16 character set

Loads the entire 8x16-pixel character set from the VGA-BIOS into one of the two character set tables.

Input

AH = 11H

AL = 04H

BL = Corresponding character set table (0 or 1)

Output No output

Remarks:

The loaded character set is not activated, nor are the CRTC registers programmed to the size of the characters. The changes will not be visible on the screen unless the character definitions are loaded into the active character table. The VGA card displays 25 text lines on the screen.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 10H

EGA/VGA

Screen: Load and activate user-defined character set

Loads a user-defined character set from RAM into one of the two EGA character tables and activates it by programming the CRTC registers.

Input

AH = 11H

AL = 10H

BH = Lines per character (also bytes per character)

BL = Character table (0 or 1)

CX = Number of characters in table

DX = ASCII code of first character in table

ES = Segment address of character table in RAM

BP = Offset address of character table in RAM

Output No output

Remarks:

A maximum of 512 characters can be loaded per character table.

The number of text lines displayed on the screen results from the height of the individual characters. It is calculated by dividing the number of screen lines (350) by the character height.

The starting and ending lines of the screen cursor are automatically adapted to the height of the new character matrix.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 11H	EGA/VGA
Screen: Load and activate 8x14 character set	

Loads the entire 8x14-pixel character set from EGA/VGA ROM-BIOS into one of the two character set tables, and activates it by programming the CRTC registers.

Input AH = 10H

AL = 11H

BL = Character table (0 or 1)

Output No output

Remarks:

The function sets the EGA screen to display 25 lines of text, or sets the VGA screen to display 28 lines of text.

The starting and ending lines of the screen cursor are automatically adapted to the height of the new character matrix.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 12H	EGA/VGA
Screen: Load and activate 8x8 character set	

Loads the entire 8x8-pixel character set from the ROM-BIOS of the EGA/VGA card into one of the two character set tables, and activates it by programming the CRTC registers.

Input AH = 10H

AL = 12H

BL = Character table (0 or 1)

Output No output

Remarks:

The function sets the screen to display 43 lines of text (EGA) or 50 lines of text (VGA).

The starting and ending lines of the screen cursor are automatically adapted to the height of the new character matrix.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 14H	VGA
Screen: Load 8x16 character set	

Loads a complete 8x16 character set from the VGA card BIOS into one of the two character set tables, and activates it through CRTIC register programming.

Input

AH = 10H

AL = 14H

BL = Character table (0 or 1)

Output No output

Remarks:

When this function is called, the VGA card displays 25 lines of text on the screen.

The starting and ending lines of the screen cursor automatically change to match the height of the new character matrix.

The contents of registers BX, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 11H, Subfunction 30H	EGA/VGA
Screen: Get information about the character generator	

Returns various information about the current status of the character generator.

Input

AH = 11H

AL = 03H

BH = Type of information desired

BH=0: contents of interrupt vector 1FH

BH=1: contents of interrupt vector 43H

BH=2: address of the ROM 8x14 character table

BH=3: address of the ROM 8x8 character table

BH=4: address of the second half of the 8x8 character table

BH=5: address of the alternative ROM 9x14 character table

BH=6: Address of the alternative ROM 8x16 character table

BH=7: Address of the alternative ROM 9x16 character table

Output	CX = Height of current character matrix
	DL = Number of columns per line - 1
	ES = Segment address of the pointer
	BP = Offset address of the pointer

Remarks:

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers CS, DS and SS are not affected by this function.

Interrupt 10H, Function 12H, Subfunction 10H	EGA/VGA
Screen: Determine EGA/VGA configuration	

Reads the configuration of the EGA/VGA card.

Input	AH = 12H
	BL = 10H
Output	BH = Monitor connected
	BH=0: color or high-resolution monitor
	BH=1: monochrome monitor
	BL = EGA/VGA RAM capacity
	BL=0: 64K
	BL=1: 128K
	BL=2: 192K
	BL=3: 256K

Remarks:

The contents of registers DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 12H, Subfunction 20H	EGA/VGA
Screen: Activate alternate hardcopy routine	

Installs an alternative hardcopy routine which prints as many lines as are displayed on the screen. The hardcopy routine of the normal ROM-BIOS always prints 25 lines and is not suited for creating a hardcopy of the EGA/VGA modes, which display more than 25 lines on the screen.

Input	AH = 12H
	BL = 20H
Output	No output

Remarks:

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 12H, Subfunction 30H	EGA/VGA
Screen: Specify number of scan lines	

Selects the number of scan lines on the screen.

Input	AH = 12H
	BL = 30H
	AL = Scan line status
	AL=0: 200 scan lines (EGA and VGA)
	AL=1: 350 scan lines (EGA and VGA)
	AL=2: 400 scan lines (VGA only)

Output	No output
---------------	-----------

Remarks:

The selected number of scan lines can only be displayed when the appropriate video card and monitor are in use. For example, a CGA monitor can only display 200 scan lines, even if the video card can operate in a higher resolution.

The contents of registers BX, CX, DX, SI, DI and BP and all segment registers are not affected by this function.

Interrupt 10H, Function 12H, Subfunction 31H	VGA
Screen: Toggle palette registers loading	

Toggles the automatic loading of palette registers in VGA-BIOS. The system either loads alternate display modes when function 00H is invoked, or loads default values.

Input	AH = 12H
	BL = 31H
	AL = Automatic palette register loading
	AL=0: Yes
	AL=1: No

Output	No output
---------------	-----------

Remarks:

The contents of registers BX, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 12H, Subfunction 32H	EGA/VGA
Screen: Enable/disable CPU access to video RAM	

Enables or disables direct CPU access to video RAM and its different I/O ports.

Input

AH = 12H

BL = 32H

AL = Access status

AL=0: Access enabled

AL=1: Access denied

Output No output

Remarks:

The EGA BIOS doesn't recognize this function, but you can still suppress video card access directly using bit 1 of the output register (port address 3C2H).

The contents of registers BX, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 12H, Subfunction 33H	VGA
Screen: Enable/disable automatic gray scaling in DAC color registers	

Toggles automatic gray scaling in VGA-BIOS. This is different from function 10H, subfunction 1BH, which enables selective gray scaling in DAC color registers.

Input

AH = 12H

BL = 33H

AL = DAC color register gray scaling

AL=0: On

AL=1: Off

Output No output

The contents of registers BX, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 12H, Subfunction 34H	VGA
Screen: Enable/disable text cursor emulation	

Toggles text cursor emulation mode. Calling function 01H (for defining the starting and ending lines of the cursor) doesn't compensate for character matrices in different resolutions. This function controls that change when in VGA mode.

Input

AH = 12H

BL = 34H

AL = Cursor emulation mode

AL=0: On

AL=1: Off

Output No output

Remarks:

The contents of registers BX, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 12H, Subfunction 36H

VGA

Screen: Suppress screen refresh

Temporarily suppresses screen refresh. Disabling refresh relieves video RAM of many system level tasks, especially those involving complex screen graphics.

Input

AH = 12H

BL = 36H

AL = Screen refresh

AL=0: On

AL=1: Off

Output No output

Remarks:

The contents of registers BX, CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 13H

EGA/VGA

Screen: Display a string

Displays a string at a specified position on the screen, in a specific display page. The characters are taken from a buffer whose address is passed to the function.

Input

AH = 13H

AL = Output mode (0-3)

AL=0: Attribute in BL, reserve cursor position

AL=1: Attribute in BL, update cursor position

AL=2: Attributes in buffer, reserve cursor position

AL=3: Attributes in buffer, update cursor position

BL = Attribute byte of characters (modes 0 and 1 only)

CX = Number of characters to be printed

DH = Screen line
 DL = Screen column
 BH = Video page
 ES = Segment address of the buffer
 BP = Offset address of the buffer

Output No output

Remarks:

In modes 1 and 3 the cursor position is placed after the last character of the string so that BIOS output will continue at the character after the string. This does not happen in modes 0 and 2.

In modes 0 and 1 the buffer contains only the ASCII codes of the characters to be printed. The color of all the characters in the string is specified by the BL register. In modes 2 and 3, each character in the buffer is followed by the corresponding attribute byte, so that each character has its own attribute. The BL register does not have to be loaded in these modes. Although the string must be twice as long as the number of characters to be printed in these modes, the CX register contains just the number of ASCII characters to be printed, not the string buffer's length.

Control codes such as bell and carriage return are interpreted as control codes and not as normal ASCII codes. An error occurs when carriage return and linefeed are printed on a display page other than zero, however. These characters may be printed on display page 0, regardless of the display page specified in BH.

When the last screen position is reached the screen will move up one line and the output will continue with the first column of the last screen line.

When printing in the graphic mode the contents of the BL register determine the foreground color of the character (the background is zero). If bit seven of the BL register is set, the color value will be XORed with the old color value.

This function can also be used to print characters in the graphic mode, in which case the character patterns will be taken from one of the EGA/VGA character tables.

The contents of registers BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 10H, Function 14H	EGA/VGA
Undocumented	

Interrupt 10H, Function 15H	EGA/VGA
Undocumented	

Interrupt 10H, Function 16H	EGA/VGA
Undocumented	

Interrupt 10H, Function 17H	EGA/VGA
Undocumented	

Interrupt 10H, Function 18H	EGA/VGA
Undocumented	

Interrupt 10H, Function 19H	EGA/VGA
Undocumented	

Interrupt 10H, Function 1AH	VGA
Screen: Determine video card type	

Determines the existence of the active video card.

Input	AH = 1AH
	AL = 0
Output	AL = 1AH
	BL = Device code for active video card
	BH = Device code for inactive video card

Remarks:

If the value 1AH is not loaded into the AL register, then the video card in operation is not a VGA card (the 1AH indicates a VGA-BIOS). The function can return the following device codes:

- FFH = Unknown video card
- 00H = No video card
- 01H = MDA with monochrome display
- 02H = CGA with CGA monitor
- 04H = EGA with EGA or multisync monitor
- 05H = EGA - monochrome display
- 07H = VGA - analog monochrome display
- 08H = VGA - analog color display (VGA, multisync)

The contents of registers CX, DX, SI, DI, BP and all segment registers are not affected by this function.

Interrupt 10H, Function 1BH	VGA
Get VGA-BIOS and video mode status	

Gets information about the current VGA-BIOS and video card status.

Input

AH = 1BH

BX = 0

ES:DI = Pointer to a buffer

Output

AL = 1BH

Remarks:

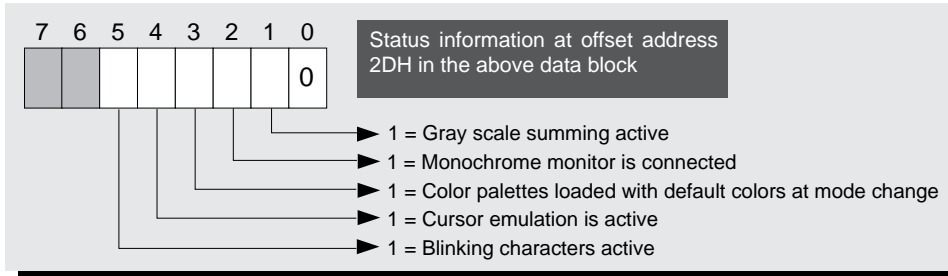
If the value 1BH is not loaded into the AL register, then the video card in operation is not a VGA card (the 1BH indicates a VGA-BIOS).

The buffer passed during the call of this function must have a minimum capacity of 64 bytes, because the VGA-BIOS stores a table there which contains information describing the current video mode.

The contents of registers BX, CX, DX, SI, DI and BP and all segment registers are not affected by this function.

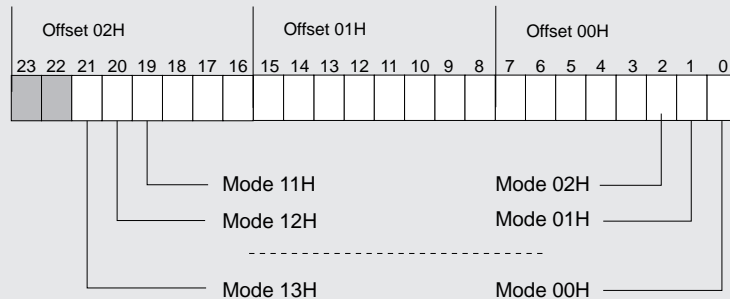
Data block with mode dependent VGA-BIOS status information		
Off	Content	Type
00H	Address of table containing static information	1 ptr
04H	Code number of current video mode	1 byte
05H	Number of displayed screen or pixel columns	1 word
07H	Length of display page in video RAM	1 word
09H	Starting address of current display page in video RAM	1 word
0BH	Cursor positions in maximum of eight display pages in column/row order	16 bytes
1BH	Ending row of cursor (pixel row)	1 byte
1CH	Starting row of cursor (pixel row)	1 byte
1DH	Number of current display page	1 byte
1EH	Port address of the CRT controller address register	1 word
20H	Current contents of CRTC control registers at port address 3B8H (MDA emulation) or 3D8H (VGA)	1 byte
21H	Current color selection register contents at port address 3D9H	1 byte
22H	Number of screen rows displayed	1 byte
23H	Height of characters in pixel rows	1 word
25H	Code number of active video adapter (see function 1AH, subfunction 00H)	1 byte
26H	Code number of inactive video adapter (see function 1AH, subfunction 00H)	1 byte
27H	Number of displayable colors (0 = monochrome)	1 word

Data block with mode dependent VGA-BIOS status information (continued)		
Off	Content	Type
29H	Number of screen pages	1 byte
2AH	Number of displayed pixel rows 0 = 200 pixel rows 1 = 350 pixel rows 2 = 400 pixel rows 3 = 480 pixel rows	1 byte
2BH	Number of character table used with characters whose	1 byte
2CH	Number of character table used with characters whose third bit of the attribute byte is 1	1 byte
2DH	Miscellaneous information #1 (see below)	1 byte
2EH	Reserved	3 bytes
31H	Size of available video RAM	1 byte
	0 = 64K 1 = 128K 2 = 192K 3 = 256K	
32H	Reserved	14 bytes
64 bytes		

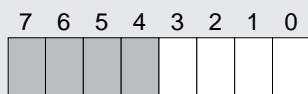


Data block with mode independent VGA-BIOS status information		
Off.	Content	Type
00H	Address of table containing static information	1 ptr
00H	Supported video modes (see below)	3 bytes
03H	reserved	4 bytes
07H	Number of pixel rows in text mode (see below)	1 byte
08H	Maximum number of character sets that can be displayed	1 byte
09H	Number of character sets loadable into video RAM	1 byte
0AH	VGA-BIOS capability information (see below)	1 byte
0BH	More VGA-BIOS capability information (see below)	1 byte
0CH	Reserved	2 bytes
0EH	Reserved	2 bytes
16 bytes		

Video mode support information found at offset address 00H as listed in the above data block

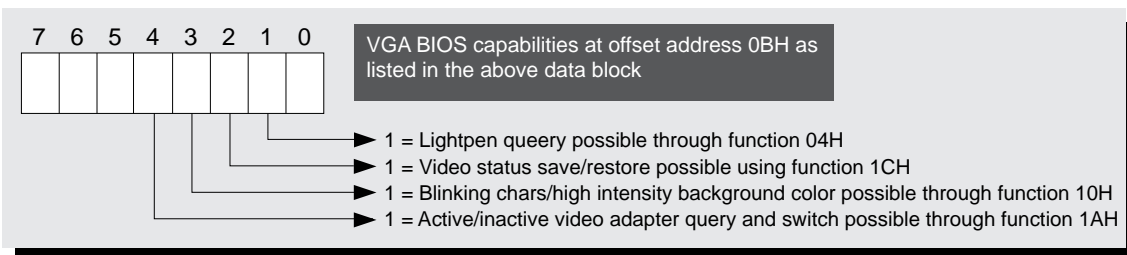
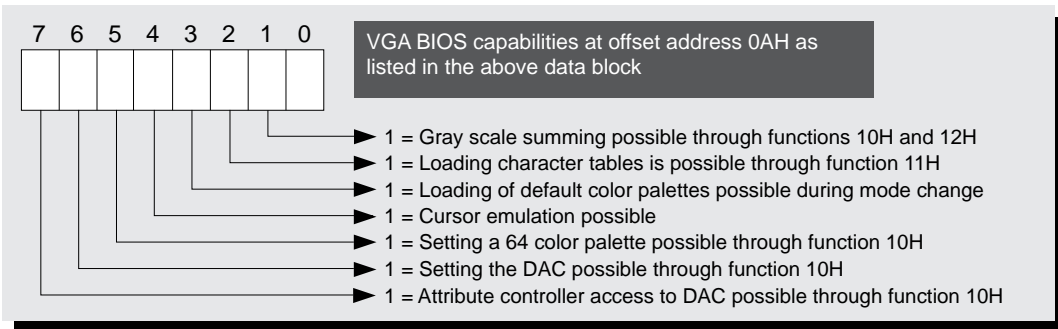


Bit contains 00000001 if corresponding video mode is supported



Pixel row information found at offset address 07H as listed in the above data block

- 1 = 200 pixel rows possible
- 1 = 350 pixel rows possible
- 1 = 400 pixel rows possible
- 1 = 480 pixel rows possible

**Interrupt 10H, Function 1CH, Subfunction 00H****VGA**

Return save/restore status

Returns information about the save/restore buffer.

Input

AH = 1CH

AL = 0

CX = Components to be saved

Bit 0=Video hardware

Bit 1=Data areas of VGA-BIOS

Output

AL = 1CH

BX = Buffer size in units of 64 bytes

Remarks:

If the value 1CH is not returned in the AL register, no VGA-BIOS exists, indicating that no VGA card is installed.

During the function call, the three lowest bits in the CX register mark the video hardware components and the VGA-BIOS which should be stored. The bit is set to 1 if the corresponding component should be stored.

Video RAM is not considered part of video hardware, and must be stored separately by programs.

The contents of the CX, DX, SI, DI and BP registers and all segment registers are not affected by this function.

Interrupt 10H, Function 1CH, Subfunction 01H	VGA
Store video status	

Stores video hardware/VGA-BIOS status received after calling subfunction 00H.

Input AH = 1CH
 AL = 1
 CX = Components to be stored
 Bit 0=Video hardware
 Bit 1=Data areas of VGA-BIOS
 ES:BX = Pointer to buffer in which information should be stored

Output AL = 1CH

Remarks:

If the value 1CH is not returned in the AL register, no VGA-BIOS exists, indicating that no VGA card is installed.

Subfunction 00H must be used to determine the buffer size required before calling this function. Notice the contents of the CX register remains unchanged between the two calls, During the function call, the three lowest bits in the CX register mark the video hardware components and the VGA-BIOS which should be stored. The bit is set to 1 if the corresponding component should be stored. Video RAM is not considered part of video hardware, and must be stored separately by programs.

The contents of the BX, CX, DX, SI, DI and BP registers and all segment registers are not affected by this function.

Interrupt 10H, Function 1CH, Subfunction 02H	VGA
Restore video status	

Restores video hardware/VGA-BIOS status stored by a preceding call to subfunction 01H.

Input AH = 1CH
 AL = 2
 CX = Components to be restored
 Bit 0=Video hardware
 Bit 1=Data areas of VGA-BIOS
 ES:BX = Pointer to buffer in which information was previously stored

Output AL = 1CH

Remarks:

If the value 1CH is not returned in the AL register, no VGA-BIOS exists, indicating that no VGA card is installed. This function call makes sense only following a call to subfunction 01H, where no more components can be restored than were stored previously. Before the function call note the content of the CX register.

The contents of the BX, CX, DX, SI, DI and BP registers and all segment registers are not affected by this function.

VESA Standard Functions

The VESA interface makes standardized functions available for access to Super VGA cards. This interface allows a program to connect with the various Super VGA cards, even though these cards are isolated from programs. The six different VESA standard functions are subfunctions of function 4FH, which links the VESA driver (or VESA compatible VGA BIOS) to BIOS interrupt 10H.

VESA BIOS Graphic Modes			
Code	Resolution	Colors	Memory
100H	640x 400	256	256K
101H	640x 480	256	512K
102H	800x 600	16	256K
103H	800x 600	256	512K
104H	1024x 768	16	512K
105H	1024x 768	256	1 Meg
106H	1280x1024	16	1 Meg
107H	1280x1024	256	1.25 Meg
6AH	800x 600	16	256 K

Interrupt 10H, Function 4FH, Subfunction 00H	VESA
Get Super VGA card information	

Reads the capabilities of the active Super VGA card, and determines which VESA functions are supported by that card

Input	AH = 4FH
	AL = 00H
	ES:DI = FAR pointer to the info buffer
Output	AL = 4FH: VESA functions are supported
	AH = 00H: VESA functions are supported

Remarks

The info buffer cited in the ES:DI registers must have 256 bytes of memory available. If the function executes correctly, it contains the following information after the call:

Structure of the Info Buffer		
Offset	Content	Type
00H	VESA signature ("VESA")	4 byte
04H	VESA version (higher level version number)	1 byte
05H	VESA version (lower level version number)	1 byte
06H	FAR pointer to ASCII string containing the name of card's manufacturer	1 dword
0AH	Flag, indicating capabilities of card (not used: Flag = 0000H)	1 dword
0EH	FAR pointer to list of code numbers indicating supported video modes	1 dword

The list with the code numbers of the supported video modes, which are passed in the last field of the buffer, consists of various words, which indicate the code of a video mode according to the table above. This list is terminated by a word containing the value 0FFFFH. The length of the list varies from card to card.

Interrupt 10H, Function 4FH, Subfunction 01H	VESA
Query data for a VESA mode	

This function returns information about a VESA mode, but does not switch it on.

Input	AH = 4FH
	AL = 01H
	CX = Code number of the desired VESA mode
	ES:DI = FAR pointer to info buffer
Output	AL = 4FH and
	AH = 00H : function was executed in an orderly manner

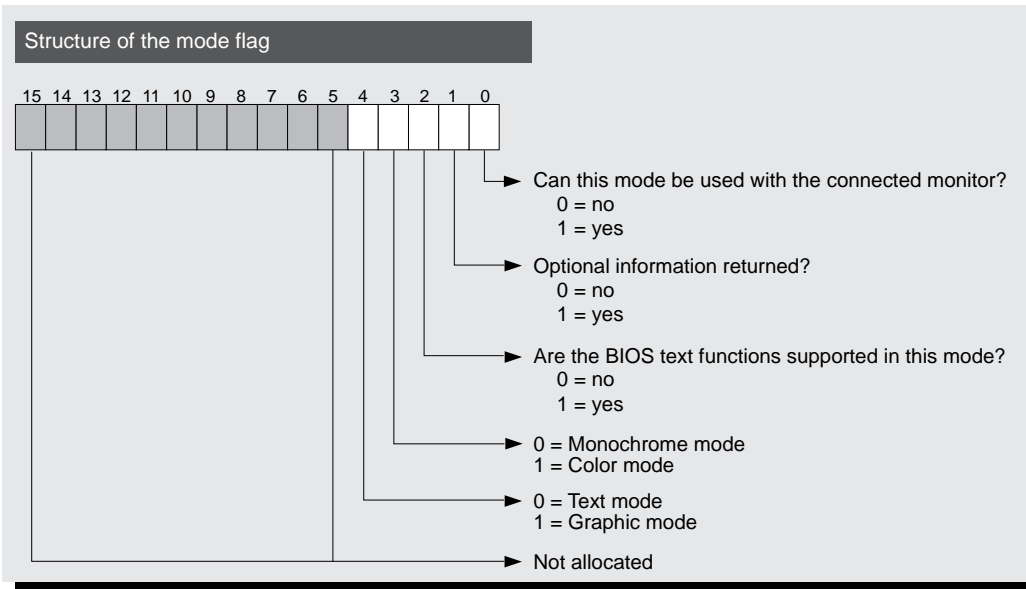
Remarks

This function should only be called, after the subfunction 00H was called successfully and therefore the existence of a VESA driver has been proven. Also only modes can be queried which appear in the mode list of function 00H.

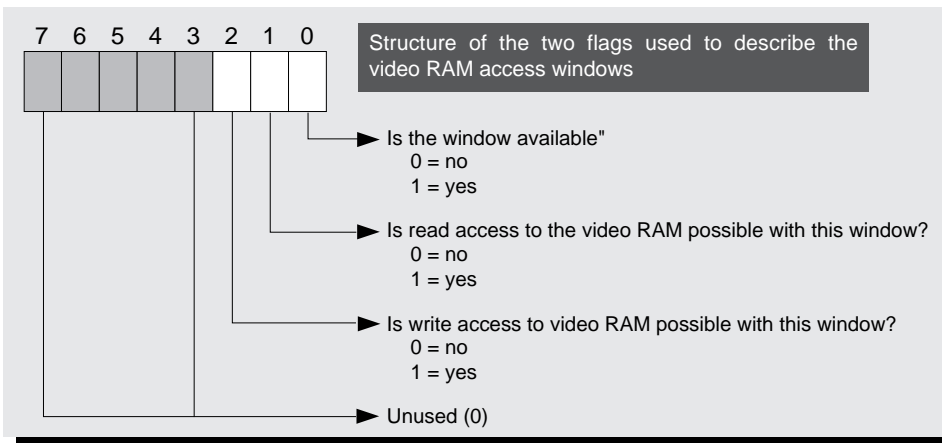
The info-buffer, whose address is passed to the function, must offer 29 bytes of storage space. If the function returns successfully to the caller, it contains the following information:

The Info-Buffer		
Offs	Content	Type
00H	Mode-Flag, see below	1 WORD
02H	Flags for the first access window, see below	1 BYTE
03H	Flags for the second access window, see below	1 BYTE
04H	Granularity in K, with which the two access windows can be moved	1 WORD
06H	Size of the two access windows in KB	1 WORD
08H	Segment address of the first access window	1 WORD
0AH	Segment address of the second access window	1 WORD
0CH	FAR-pointer to the routine for setting of the visible area in the two access windows	1 DWORD
10H	Number of bytes, occupied by the individual dot line in Video-RAM	1 WORD
Optional information, see mode-Flag		
12H	X-resolution in dots/characters	1 WORD
14H	Y-resolution in dots/characters	1 WORD
16H	Width of the der character matrix in dots	1 BYTE
17H	Height of the character matrix in dots	1 BYTE
18H	Number of Bit-Planes	1 BYTE
19H	Number of bits per screen dot	1 BYTE
1AH	Number of storage blocks	1 BYTE
1BH	Storage model	1 BYTE
1CH	Size of the storage blocks in KB	1 BYTE

The mode flag (offset 00H) tells if the optional fields in the info buffer of the VESA function were filled out and returns important information about the desired mode.



The two access windows (offset 02H and 03H) are described through the following bit fields:



The storage model (offset 1BH) reflects the structure of the Video RAM in the desired video mode. These codes are known.

Valid Codes for the description of the storage model			
Num.	Task	Num.	Task
00H	Text mode	04H	Packed format with two dots with 4 bits per byte
01H	CGA-Format, i.e. 2 or 4 storage blocks	05H	Normal EGA-/VGA format for 256 color graphic mode
02H	Hercules-Format with 4 storage blocks	06H-0FH	Reserved
03H	Normal EGA-/VGA format for 16 color graphic mode	10H-FFH	Manufacturer specific code, not used until now

Interrupt 10H, Function 4FH, Subfunction 02H	VESA
Switch on VESA mode	

A VESA mode can be switched on by using this function.

Input AH = 4FH
 AL = 02H
 BX = Code number of the desired mode

Output AL = 4FH and
 AH = 00H: function was executed in an orderly manner

Remarks

This function should only be called, after the subfunction 00H was called successfully and therefore the existence of a VESA driver has been proven. Also only modes can be queried which appear in the mode list of function 00H.

Bit 15 in the code number of the video mode in the BX register, can be set when the Video RAM should not be erased during the initialization of the video mode.

Interrupt 10H, Function 4FH, Subfunction 03H	VESA
Query current mode	

This function returns the code number of the current video mode and takes into consideration also the non VESA modes.

Input AH = 4FH
 AL = 03H

Output AL = 4FH and
 AH = 00H : was executed in an orderly manner in this case
 BX = Code-number of the current mode

Remarks

This function should only be called, after the subfunction 00H was called successfully and therefore the existence of a VESA driver has been proven.

Interrupt 10H, Function 4FH, Subfunction 04H/00H	VESA
Determine the size of the safety buffer	

For the call of the subfunction 04H/01H which follows, this function is used to determine the size of the required safety buffer.

Input AH = 4FH
 AL = 04H
 DL = 00H
 CX = Components of the video-status to be stored

Output

AL = 4FH and

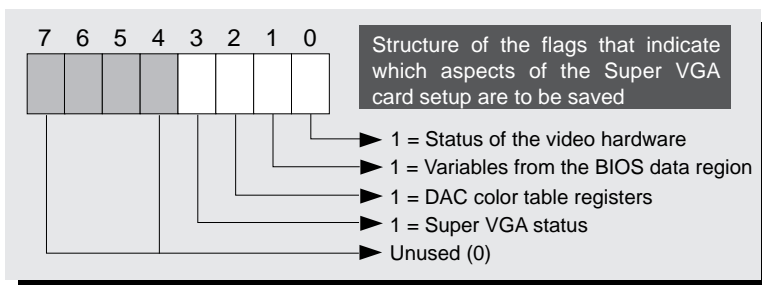
AH = 00H : Function was executed in an orderly manner in this case

BX = Number of consecutive 64 byte blocks which are required as safety buffer

Remarks

This function must be called before the subfunction 04H/01H to determine the size of the safety buffer which is made available to the subfunction 04H/01H.

During the function call, the various bits in the CX register indicate, which components of the video-status should be stored. Only the low byte of the CX register is used, in which the following bits are of significance:

**Interrupt 10H, Function 4FH, Subfunction 04H/01H****VESA**

Store Video-Status of the Super VGA card

After a preceding call of the subfunction 04H/00H, the desired information about the various components of the video status are stored in a buffer of the caller by this function.

Input

AH = 4FH

AL = 04H

DL = 01H

CX = Components of the video status to be stored

ES:BX = FAR pointer to safety buffer

Output

AL = 4FH and

AH = 00H: Function was executed in an orderly manner

Remarks

The buffer that was passed must have the size which was indicated through the BX register, during the preceding call of the subfunction 04H/00H.

The various bits in the CX register indicate during the function call which components of the video status should be stored. See the subfunction 04H/00H.

Interrupt 10H, Function 4FH, Subfunction 04H/02H	VESA
Restore the video status of the Super VGA card	

After the storage of the video-status with the help of the subfunction 04H/01H, this status can be restored again through the call of this function.

Input

- AH = 4FH
- AL = 04H
- DL = 01H
- CX = Components of the video-status to be restored
- ES:BX = FAR pointer to the segment address of the safety buffer

Output

- AL = 4FH and
- AH = 00H: Function was executed in an orderly manner

Remarks

The buffer which was passed, must have been loaded previously with information about the video status, through a call of the subfunction 04H/01H.

The various bits in the CX register indicate during the function call, which components of the video status should be restored. See the subfunction 04H/00H.

Interrupt 10H, Function 4FH, Subfunction 05H/00H	VESA
Determine access window in the video RAM	

This function is used to include a certain part of the video RAM in one of the two VESA access windows and thereby make it addressable for a program.

Input

- AH = 4FH
- AL = 05H
- BH = 00H
- BL = Access window (0 or 1)
- DX = Start address

Output

- AL = 4FH and
- AH = 00H : Function was executed in an orderly manner

Remarks

The start address in the DX-Register should be viewed in relation to the granularity of the window, which can be determined by the call of the subfunction 01H.

The second access window can only be addressed with this function, if a preceding call of the subfunction 00H has indicated, that two access windows actually exist.

Interrupt 10H, Function 4FH, Subfunction 05H/01H	VESA
Query access window on the video RAM	

By using this function, the location of the access window in relation to the video RAM of the Super VGA card can be queried.

Input

AH = 4FH

AL = 05H

BH = 01H

BL = Access window (0 or 1)

Output

AL = 4FH and

AH = 00H: Function was executed in an orderly manner

DX = Starting address

Remarks

The start address in the DX register should be viewed in relation to the granularity of the window, which can be determined by the call of the subfunction 01H.

DOS API Interrupts And Functions

More than 100 functions can be accessed using Interrupt 21h, which are made available to a program by DOS and therefore are designated as the Application-Program-Interface (DOS-API).

These functions are described in this chapter, including a whole series of functions whose significance were never officially publicized by Microsoft and are therefore designated as undocumented. The "undocumented" functions which are described here, have been already used in many thousands of commercial applications and Microsoft can't afford to leave them out of the API in a future DOS version. If you find no official function to help you in some application, you may use one of these undocumented functions.

Interrupt 20H	DOS (Version 1.0 and above)
Terminate program	

Restores the three interrupt vectors whose contents were stored in the PSP before the program call, terminates the currently running program and returns control to MS-DOS. If the program redirected the vectors to its own routine, these vectors cannot be overwritten by another program. However, the terminating program releases the RAM it had occupied. Before turning control over to the calling program, this memory releases and all data buffers clear.

Input CS = Segment address of the PSP

Output No output

Remarks

COM programs automatically store the segment address of the PSP in the CS register. EXE programs require additional programming to load the segment address of the PSP into the CS register. Since the code and the PSP are stored in two separate segments, the address of the PSP must be loaded into the CS register. The code executes from another segment, which makes it impossible to call interrupt 32. To help overcome this problem, the value 0 and then the segment address of the PSP are pushed onto the stack. If a FAR RETURN command then executes, the program execution continues in the PSP segment at offset address 0. There a call for interrupt terminates the program.

For the first version of DOS, this interrupt is the usual method for ending a program. To terminate a program in DOS Version 2 and up, functions 31H or 4CH of DOS interrupt 21 H should be called instead.

Interrupt 21H, Function 00H	DOS (Version 1.0 and above)
Terminate program	

Terminates execution of the currently running program and returns control to the calling program. Before this happens, the three interrupt vectors, whose contents had been stored in the PSP before the call of the program, are restored. If the program redirects these vectors to its own routine, they cannot be overwritten by another program. However, the terminating program does release the RAM it had occupied. Before turning control over to the calling program, the function releases this memory and clears all buffers.

Input AH = 00H
CS = segment address of the PSP

Output No output

Remarks

COM programs automatically store, in the CS register, the segment address of the PSP. Since the code and the PSP are stored in two separate segments, you cannot execute this function from an EXE program.

Instead of this function, use either function 31H or 4CH of interrupt 21H for terminating a program.

Interrupt 21H, Function 01H	DOS (Version 1 and above)
Character input with echo	

Reads a character from the standard input device and displays it on the standard output device. When the function is called but a character doesn't exist, the function waits until a character is available. Since standard input and output can be redirected, this function is able to read a character from an input device other than the keyboard and send it to an output device other than the screen. The characters that are read may originate from other devices or from a file. If the character comes from a file, the input doesn't redirect to the keyboard once it reaches the end of the file. So, the function continues to try to read data from the file after it passes the end.

Input AH = 01H

Output AL = Character read

Remarks

If extended key codes are read, the function passes code 0 to the AL register. The function must be called again to read the actual code.

If the function encounters a **Ctrl C** character (ASCII code 3), it calls interrupt 23H.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 02H	DOS (Version 1 and above)
Character output	

Displays a character on the standard output device. Since this device can be redirected, the character can be displayed on another output device or sent to a file. This function doesn't test whether or not the storage medium (disk or hard disk) is already full. Therefore, it will continue to try to write characters to this file.

Input AH = 02H

DL = code of the character to be output

Output No output

Remarks

Control codes such as backspace, carriage return and linefeed are executed when the function sends characters to the screen. If the output is redirected to a file, control codes are stored as normal ASCII codes.

If the function encounters a **Ctrl****C** character (ASCII code 3), it calls interrupt 23H.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 03H**DOS (Version 1 and above)**

Read character auxiliary input

Reads a character from the serial port. Access defaults to the device with the designation COM1, unless a MODE command previously redirected serial access.

Input AH = 03H**Output** AL =Character received*Remarks*

Since the serial port has no internal buffer, it can receive characters faster than it can read them. The unread characters are then ignored.

Before calling this function, communication parameters (baud rate, number of stop bits, etc.) must be set using the MODE command. Otherwise DOS defaults to 2400 baud, one stop bit, no parity and a word length of 8 bits.

The BIOS functions called from interrupt 14H are a more efficient way to access the serial port. Since they also allow reading of the serial port status, these functions offer more flexibility than the DOS functions.

If the function encounters a **Ctrl****C** character (ASCII code 3), it calls interrupt 23H.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 04H**DOS (Version 1 and above)**

Auxiliary output

Sends a character to the serial port. Unless a MODE command previously redirected serial access, access defaults to the device with the designation COM1.

Input AH = 04H

DL =Character set for output

Output No output*Remarks*

As soon as the receiving device sends a signal to the function indicating that it is ready to receive it, the function transmits the character. Control then returns to the calling program.

Before calling this function, communication parameters (baud rate, number of stop bits, etc.) must be set using the MODE command. Otherwise DOS defaults to 2400 baud, one stop bit, no parity and a word length of 8 bits.

The BIOS functions called from interrupt 14H are a more efficient way to access the serial port. Since they also allow reading of the serial port status, they offer more flexibility than the DOS functions.

If the function encounters a **Ctrl****C** character (ASCII code 3), it calls interrupt 23H.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 05H
DOS (Version 1 and above)

Character output to printer

Sends a character to the printer. Access defaults to the device with the designation LPT1 (identical to PRN), unless a MODE command previously redirected printer access.

Input

AH = 05H

DL = Character code to be printed

Output

No output

Remarks

The function transmits the character only when the printer signals that it is ready to receive it. Then control returns to the calling program.

If the function encounters a **Ctrl C** character (ASCII code 3), it calls interrupt 23H.

The BIOS functions called from interrupt 17H are more efficient for printer access. They offer more flexibility than the DOS printer functions for character output.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 06H
DOS (Version 1 and above)

Direct console I/O

Reads characters from the standard input device and displays them on the standard output device. The read or written character isn't tested by the operating system (e.g., **Ctrl C** has no effect on the program). Since standard input and output can be redirected, this function can read a character from an input device other than the keyboard and sends it to an output device other than the screen. The characters read may originate from other devices or from a file. When writing characters, this function doesn't test whether the storage medium (disk or hard disk) is already full. Also, the calling program cannot determine whether all the characters have been read from an input file.

During character input, the function doesn't wait until a character is available. Instead, the function returns control to the calling program.

Input

AH = 06H

DL = 0-254: Send character code

DL = 255: Read a character

Output

Character output: No output

Character input: Zero flag=1: No character ready

Zero flag=0: Character read is in the AL register

Remarks

If extended key codes are read, the function passes code 0 to the AL register. The function must be called again to read the actual code.

ASCII code 255 (blank) cannot be displayed with this function because the function interprets ASCII code 255 as a command to input a character.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 07H**DOS (Version 1 and above)**

Unfiltered character input without echo

Reads a character from the standard input device without displaying the character on the standard output device. If a character doesn't exist when the function is called, the function waits until a character is available. The read character is not tested by the operating system (e.g., **Ctrl****C** has no effect on the program). Since standard input and output can be redirected, this function can read a character from an input device other than the keyboard. The characters that are read may originate from other devices or from a file. If the characters come from a file, the input doesn't redirect to the keyboard once it reaches the end of file. This causes the function to continue to try reading data from the file after it passes the end of file.

Input AH = 07H**Output** AL =Character read*Remarks*

If extended key codes are read, the function passes code 0 to the AL register. The function must be called again to read the actual code.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 08H**DOS (Version 1 and above)**

Character input without echo

Reads a character from the standard input device without displaying the character on the standard output device. If no character exists when the function is called, the function waits until a character is available.

Since standard input can be redirected, this function can read a character from an input device other than the keyboard. The characters read may originate from other devices or from a file. If the characters come from a file, the input doesn't redirect to the keyboard on reaching the end of file, so the function continues to try reading data from the file after it passes the end of file.

Input AH = 08H**Output** AL =Character read*Remarks*

If extended key codes are read, the function passes code 0 to the AL register. The function must be called again to read the actual code.

If the function encounters a **Ctrl****C** character (ASCII code 3), it calls interrupt 23H.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 09H**DOS (Version 1 and above)**

Output character string

Displays a character string on the standard output device. Since this device can be redirected, the character may be displayed on another output device or sent to a file. This function doesn't test whether or not the storage medium (disk or hard disk) is already full, and will continue to try to write the string to a file.

Input

AH = 09H

DS = String segment address

DX = String offset address

Output

No output

Remarks

The string must be stored in memory as a series of bytes which contain the ASCII codes of the characters to be output. A dollar sign character "\$" (ASCII code 36) indicates, to DOS, the end of the string.

Control codes, such as backspace, carriage return and linefeed, are executed within the string.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 0AH**DOS (Version 1 and above)**

Buffered input

Reads a number of characters from the standard input device and transmits the characters to a buffer. The input ends when the user presses the **Enter** key. The ASCII code of this key (13) is then placed in the buffer as the last character of the string.

Since standard input can be redirected, this function can read a character from an input device other than the keyboard. The characters read may originate either from other devices or from a file. If the characters come from a file, the input doesn't redirect to the keyboard on reaching the end of file, so the function continues to try reading data from the file after it passes the end.

Input

AH = 0AH

DS = Buffer segment address

DX = Buffer offset address

Output

No output

Remarks

The first byte of the buffer accepts the maximum number of characters (including the carriage return which ends the input) which can be read into the buffer, starting at memory location 2. In order to inform the function of the maximum number of characters it may read, this information must be entered, by the calling program, into the buffer before the function call.

After completion of the input, DOS places the number of characters read (excluding the carriage return) in memory location 1.

The buffer must be the number of the characters to be read plus 2 bytes.

When the input reaches the second to last memory location in the buffer, the computer beeps if you attempt to enter any character other than the **Enter** key (end of input).

Extended key codes occupy two bytes in the buffer. The first byte contains the code 0, and the second byte contains the extended key code.

If the function encounters a **Ctrl C** character (ASCII code 3), it calls interrupt 23H.

The **Backspace** and cursor keys let you edit the input without storing these keys in the buffer.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 0BH

DOS (Version 1 and above)

Get input status

Determines whether a character is available for reading from the standard input device.

Input AH = 0BH

Output AL = 0: No character available

AL = 255: One or more characters available for reading

Remarks

If the function encounters a **Ctrl C** character (ASCII code 3), it calls interrupt 23H.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 0CH

DOS (Version 1 and above)

Reset input buffer and then input

Clears the input buffer then calls one of the character input functions. Since all the character input functions get their characters from the standard input device and standard input may be redirected, this function only operates when the keyboard is the standard input device. In this case the characters could be entered before the function call but not read by a function. These existing characters are erased to ensure that the function call only reads characters which were inputted after its call.

Input AH = 0CH

AL = Function to be called during call of function 10

DS = Input buffer segment address

DX = Input buffer offset address

Output Functions 1, 6, 7 and 8: AL = Character to be read

Function 10: No output

Remarks

Functions 1, 6, 7, 8 and 10 can be passed to the function as calling functions.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 0DH	DOS (Version 1 and above)
Disk reset	

Sends all data stored in an internal DOS buffer to a block driver device (e.g., disk drive, hard disk). The open files (handles or FCBs) remain open.

Input AH = 0DH

Output No output

Remarks

Despite this function call, all open files must be closed in an orderly manner. Otherwise the current directory entry of the file may not update properly, which prevents access to new file data.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 0EH	DOS (Version 1 and above)
Select default disk drive	

Defines the the current default disk drive. Its designation appears as a prompt on the screen when the command interpreter expects input from the user. The drive indicated here will be used for all file access in which no special device was specified.

Input AH = 0EH

DL = Drive number

Output AL = Number of installed drives or volumes

Remarks

Drive A: has code number of 0, drive B: code number 1, etc.

Even if the PC has only one disk drive and one hard disk, the number of volumes in the AL register can be greater than two because the hard drive can be divided into multiple volumes. In addition, the PC can have one or more RAM disks as part of its configuration. For a PC with a single disk drive, you can only have two volumes because drive A: also simulates drive B:.

Unlike DOS Version 2, which permits 63 different device codes, DOS Version 3 permits 26 different devices (the letters A to Z). To keep compatibility between versions, limit your device access to a maximum of 26 devices.

BIOS interrupt 11H does a better job of reading the number of disk drives than this function.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 0FH	DOS (Version 1 and above)
Open file (FCB)	

Opens a file if one is available. After this function call executes successfully, the file can be read or written.

Input AH = 0FH

DS =FCB segment address of the file

DX = FCB offset address of the file

Output AL =0: File found and opened

AL =255: File not found

Remarks

Both normal and extended FCBs can be used.

If the file was found, DOS enters, into the FCB, the file size, the date and the time of its creation or last modification.

DOS sets the record length at 128 bytes. This record length can be changed in the FCB before opening a file. If you need a longer record length, the DTA must be moved (the original DTA is only 128 bytes long).

If random file access is performed, the random record field in the FCB must be set after the file opens successfully.

The file pointer points to the first byte of the file after the file opens.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 10H

DOS (Version 1.0 and above)

Close file (FCB)

Writes all data currently in the DOS buffer to the file and closes the file. In addition, the directory entry changes to reflect the new file size and the date and time of the most recent modification to the file.

Input AH = 10H

DS =FCB segment address of the file

DX = FCB offset address of the file

Output AL =0: File closed and directory entry revised

AL =255: File not found in directory

Remarks

Only open files can be closed.

For disk files, the disk which was in the drive when the function call occurred must also be the disk that contains the file. Otherwise, the function call writes an incorrect FAT and an incorrect directory to the disk, which makes the data that is already on the disk useless.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 11H

DOS (Version 1 and above)

Search for first match (FCB)

Searches for the first occurrence in the disk directory of the filename indicated in the FCB.

Input AH = 11H

DS = FCB segment address

DX = FCB offset address

Output AL = 0: File found

AL = 255: File not found

Remarks

The FCB passed to the function contains the drive specifier and the filename for which the function should search.

The filename can contain the wildcard "?" to search for a group of files.

The search is made only in the current directory of the indicated device.

If the function searches for a normal file, a normal FCB can pass the information to the function. However, if you wish to search for a file with special attributes (volume name, subdirectories, hidden files, etc.), extended FCBs must be used.

If a file was found, the DTA contains an FCB of the same type as the FCBs. This FCB in the DTA contains the found filename. For this reason, the DTA must always be large enough to accept either a normal or an extended FCB.

The DTA can be switched to its own buffer using function 1AH, to ensure that it is large enough to accept the FCB.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 12H

DOS (Version 1 and above)

Search for next match (FCB)

Searches for additional occurrences in the disk directory of the filename indicated in the FCB, after the file was found by function 17 (see above).

Input AH = 12H

DS = FCB segment address

DX = FCB offset address

Output AL = 0: File found

AL = 255: File not found (no other files available)

Remarks

This function can only be called after calling function 11H.

The FCB passed to the function contains the drive specifier and the filename for which the function should search.

If another filename was found its name is recorded in the FCB at the beginning of the DTA.

The DTA can be switched with function 1AH to its own buffer to ensure that it is large enough to accept the FCB.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 13H**DOS (Version 1 and above)**

Delete file (FCB)

Erases one or more files in the current directory of the specified device.

Input

AH = 13H

DS = FCB segment address

DX = FCB offset address

Output

AL = 0: file(s) erased

AL = 255: No file(s) found, or file(s) assigned Read-Only attribute

Remarks

The FCB passed to the function contains both the device on which the files to be erased are located and the name of the file.

The filename can contain the wildcard "?" to erase a group of files.

Only files in the current directory of the indicated device may be erased.

If the function is used to delete a normal file, a normal FCB can pass the information to the function. However, if you want to delete a file with special attributes (volume name, subdirectories, hidden files, etc.), extended FCBs must be used.

Volumes may be deleted with this function, however, subdirectories may not.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 14H**DOS (Version 1 and above)**

Sequential read (FCB)

Reads the next sequential data block from a file.

Input

AH = 14H

DS = FCB segment address

DX = FCB offset address

Output

AL = 0: Block read

AL = 1: End of file reached

AL = 2: Segment wrap

AL = 3: Partial record read

Remarks

The function can only be called after the file was opened by the indicated FCB.

The DTA reads the block. If the DTA is not large enough, function 1AH must move the DTA into its own buffer.

The FCB records the size of the block and the corresponding number of bytes read.

Error 2 occurs when the DTA reaches the end of a segment and the block being read extends beyond the end of the segment.

Error 3 occurs when a partial block appears at the end of the file. The block is read in anyway and blank spaces bring the block up to the allocated block size.

After reading a block, the file pointer resets to the beginning of the next block so that the next function call automatically reads the next block.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 15H
DOS (Version 1 and above)

Sequential write (FCB)

Writes a sequential block to a file.

Input

AH = 15H

DS = FCB segment address

DX = FCB offset address

Output

AL = 0: Block written

AL = 1: Medium (disk/hard drive) full

AL = 2: Segment overflow

Remarks

The function can only be called after the file was opened by the indicated FCB.

The DTA writes the block it contains to the file. If the DTA is not large enough to hold the file, function 1AH must be used to move the DTA into its own buffer.

The FCB records the size of the block and the corresponding number of bytes written.

Error 2 occurs if the DTA reaches the end of a segment and the block being written extends beyond the end of the segment.

After writing a block, the file pointer resets to the beginning of the next block, so that the next function call automatically writes the next block.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 16H
DOS (Version 1 and above)

Create or truncate file (FCB)

Creates a new file, or dumps the contents of an existing file (file size=0 bytes). This function call allows other functions to read or write to the open file.

Input

AH = 16H

DS = FCB segment address

DX = FCB offset address

Output AL =0: File created or cleared

AL =255: File could not be created (e.g., directory full)

Remarks

The contents of an existing file called by this function are lost.

After calling this function, the file is already open; you don't need to open the file using function 0FH (see above).

If you open the file using an extended FCB, you can assign certain attributes to the file (e.g., volume name, hidden file, etc.).

You cannot create a subdirectory using this function.

After opening the file, the file pointer moves to the first byte of the file.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 17H

DOS (Version 1 and above)

Rename file (FCB)

Renames one or more files in the current directory of the specified device.

Input AH = 17H

DS =FCB segment address

DX = FCB offset address

Output AL =0: File(s) renamed

AL =255: No file found, or new filename matches old filename

Remarks

The FCB here is a special FCB, based on a normal FCB. The first 12 bytes contain the drive specifier and the name of the file to be renamed. However, this type of FCB has the new drive specifier and the new filename stored starting at memory location 10H. The drive specifier must be identical for both filenames.

The name of the file to be renamed can contain the wildcard "?", which renames several files. If the new filename contains the wildcard "?", the places in the filename and extension where a question mark appears in this parameter remain unchanged.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 18H

DOS (Version 1 and above)

Reserved for internal use

Interrupt 21H, Function 19H

DOS (Version 1 and above)

Get default disk drive

Returns the drive specifier of the default (current) disk drive.

Input AH = 19H

Output AL = Drive specifier

Remarks

This function identifies drive A as code 0, drive B as code 1, etc.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 1AH

DOS (Version 1 and above)

Set DTA address

Transfers the DTA (Disk Transfer Area) to another area of memory. The DTA acts as buffer memory for all FCB supported file accesses.

Input AH = 1AH

DS = New DTA segment address

DX = New DTA offset address

Output No output

Remarks

This function must be called if the existing DTA has insufficient memory to handle the transmitted data.

When the program starts, MS-DOS places the DTA at address 128 in the PSP. Since the program starts after address 255 of the PSP, it is 128 bytes long.

DOS does not test the length of the DTA. Instead it assumes that the DTA is large enough to accept the transmitted data. If this is not the case, a DOS function can overwrite the excess data.

DOS recognizes an error during various functions if the DTA is at the end of a segment and the data to be transmitted exceeds the end of the segment.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 1BH

DOS (Version 1 and above)

Get allocation information for default drive

Returns information about the format of the default drive.

Input AH = 1BH

Output AL = Number of sectors per cluster

DS = Media descriptor segment address

BX = Media descriptor offset address

DX = Number of clusters

Remarks

The media descriptor can return the following codes:

F8H: Hard disk

F9H: Disk drive: double-sided, 15 sectors per track (AT only)

FCH: Disk drive: single-sided, 9 sectors per track

FDH: Disk drive: double-sided, 9 sectors per track

FEH: Disk drive: single-sided, 8 sectors per track

FFH: Disk drive: double-sided, 8 sectors per track

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 1CH

DOS (*Version 1 and above*)

Get allocation information for specified drive

Returns information about the format of the specified drive.

Input

AH = 1CH

DL = Drive specifier

Output

AL = Number of sectors per cluster

DS = Media descriptor segment address

BX = Media descriptor offset address

DX = Number of clusters

Remarks

This function identifies drive A as code 0, drive B as code 1, etc.

The media descriptor can return the following codes:

F8H: Hard drive

F9H: Disk drive: double-sided, 15 sectors per track (AT only)

FCH: Disk drive: single-sided, 9 sectors per track

FDH: Disk drive: double-sided, 9 sectors per track

FEH: Disk drive: single-sided, 8 sectors per track

FFH: Disk drive: double-sided, 8 sectors per track

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 1DH

DOS (*Version 1 and above*)

Reserved

Interrupt 21H, Function 1EH

DOS (*Version 1 and above*)

Reserved

Interrupt 21H, Function 1FH	DOS (Version 1 and above)
Get DPB pointer to current drive	

Gets the pointer to a DOS Parameter Block for the current disk drive.

Input AH = 1FH

Output AL = 000H: Operation successful
 AL = 0FFH: Error
 BX:DS = FAR pointer to DPB structure

Remarks

If the device indicated is a diskette drive, DOS accesses the drive to fill the DPB with data about the drive and the format of the diskette. For hard drives this information is already stored in the memory and no hardware access is required.

An error during the function call can only occur if an invalid drive code was indicated. If this occurs, error code 15 is returned in the AL register.

The construction of the DOS Parameter Block varies between DOS versions.

The contents of the AH, CX, DX, SI, DI, BP, CS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 20H	DOS (Version 1 and above)
Reserved	

Reads a specified file record into the DTA.

Input AH = 21H
 DS = FCB segment address
 DX = FCB offset address

Output AL = 0: Record read
 AL = 1: End of file reached
 AL = 2: Segment overflow
 AL = 3: Partial record read

Remarks

The function can only be called after the file was opened by the indicated FCB.

The record whose address is stored in the FCB starting at location 21H is read.

The DTA reads the record. If the DTA is not large enough, function 1AH must be called to move the DTA into its own buffer.

The FCB records the size of the record and the corresponding number of bytes read.

During the function call, the file pointer moves to the beginning of the record being read so that a subsequent call of a sequential read (function 14H-see above) reads the same record sequentially.

The record number does not increment following the function call, so a new call of this function would read the same record.

Error 2 occurs when the DTA reaches the end of a segment and the record being read extends beyond the end of the segment.

Error 3 occurs when a partial record appears at the end of the file. The record is read in anyway and blank spaces bring the record up to the allocated record size.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 22H

DOS (Version 1 and above)

Random write (FCB)

Writes data from memory to the specified record in a file.

Input

AH = 22H

DS = FCB segment address

DX = FCB offset address

Output

AL = 0: record was written

AL = 1: Medium (disk/hard drive) full

AL = 2: segment overflow

Remarks

The function can only be called after the file was opened by the indicated FCB.

The record whose address is stored in the FCB starting at location 21H is read.

The record is written from the DTA to the file. If the DTA is not large enough, function 1AH must move the DTA into its own buffer.

The FCB records the size of the record and the number of bytes read.

During the function call, the file pointer moves to the beginning of the record being read. This instructs subsequent calls of a sequential read (function 14H-see above) to read the same record sequentially.

The record number does not increment following the function call, so a new call of this function would read the same record.

Error 2 occurs when the DTA reaches the end of a segment and the record being written extends beyond the end of the segment.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 23H

DOS (Version 1 and above)

Get file size in records (FCB)

Determines the size of a file based on the number of records in that file.

Input

AH = 23H

DS = FCB segment address

DX = FCB offset address

Output AL =0: Number of records found starting at FCB address 21H

AL =255: File not found

Remarks

The FCB passed contains the drive specifier as well as the name and extension of the file to be examined.

Unlike the other FCB supported file accesses, the FCB requires the record size before the application can call this function.

A record size of 1 returns the size of the file in bytes.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 24H	DOS (Version 1 and above)
Set random record number	

Sets the record number in the FCB to the current position of the file pointer. Random access may begin at the point at which earlier sequential accesses left off.

Input AH = 24H
 DS =FCB segment address
 DX = FCB offset address

Output No output

Remarks

The function can only be called after the file was opened by the indicated FCB.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 25H	DOS (Version 1 and above)
Set interrupt vector	

Sets any interrupt vector to another routine.

Input AH = 25H
 AL = Interrupt number
 DS = New interrupt routine segment address
 DX = New interrupt routine offset address

Output No output

Remarks

Before calling this function, the old contents of the interrupt vector to be changed should be read and stored using function 35H. After the program terminates, the old contents of the interrupt vector should be restored.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 26H**DOS (Version 1 and above)**

Create PSP

Copies the PSP (program segment prefix) of the executing program to a specified address in memory.

Input

AH = 26H

DX = New PSP segment address

Output

No output

Remarks

The new PSP offset address is 0.

DOS Version 1 uses this function to execute other programs by creating a PSP, loading the program after this PSP and executing it.

For DOS Version 2 up, use the EXEC function 4BH to load and execute additional programs instead of this function.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 27H**DOS (Version 1 and above)**

Random block read (FCB)

Reads one or more sequentially stored records into memory.

Input

AH = 27H

CX = Number of records to be read

DS = FCB segment address

DX = FCB offset address

Output

AL = 0: Record read

CX = Number of records read

AL = 1: End of file reached

AL = 2: Segment overflow

AL = 3: Partial record read

Remarks

The function can only be called after the file was opened by the indicated FCB.

The starting record is the record whose address is stored in the FCB, starting at location 21H.

The record data passes to the DTA. If the DTA is not large enough, function 1AH must move the DTA into its own buffer.

The FCB records the size of the record and the corresponding number of bytes read.

After the function call, the file pointer moves to the end of the last record that was read so that it points to the next record (following the last record read).

Error 2 occurs when the DTA reaches the end of a segment and the record being read extends beyond the end of the segment.

Error 3 occurs when a partial record appears at the end of the file. The record is read in anyway and blank spaces bring the record up to the allocated record size.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 28H	DOS (Version 1.0 and above)
Random block write (FCB)	

Writes one or more records in sequence to the specified file.

Input

AH = 28H

CX = Number of records to be written

DS = FCB segment address

DX = FCB offset address

Output

AL = 0: Record written

CX = Number of records written

AL = 1: Medium (disk/hard drive) full

AL = 2: Segment overflow

Remarks

The function can only be called after the file was opened by the indicated FCB.

The starting record is the record whose address is stored in the FCB starting at location 21H.

The FCB records the size of the record and the corresponding number of bytes read.

The data is written from the DTA to the file. If the DTA is not large enough, function 1AH must move the DTA into its own buffer.

After the function call, the file pointer moves to the end of the last record written so that it points to the next record, which follows the last record written. The record number increments by the number of records written.

Error 2 occurs when the DTA reaches the end of a segment and the record being written extends beyond the end of the segment.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 29H	DOS (Version 1 and above)
Parse filename to FCB	

Transfers an ASCII format filename into the proper fields of an FCB. The filename can include a drive specifier, filename and file extension.

Input

AH = 29H

DS = Segment address of filename in memory

SI = Offset address of filename in memory

ES = FCB segment address

DI = FCB offset address

AL = Transmission parameters:

Bit 1 = 1: The drive specifier in the FCB changes only if the filename passed contains a drive specifier

0: The drive specifier changes anyway. If the filename passed contains no drive specifier, the the FCB defaults to 0 (current drive)

Bit 2 = 1: The filename in the FCB changes only if the filename parameter passed contains a filename

0: The filename changes. If the filename passed does not contain a filename, the filename in the FCB fills with spaces (ASCII code 32)

Bit 3 = 1: The file extension in FCB changes only if the filename passed contains an extension

0: The file extension in the FCB changes. If the filename passed has no extension, the extension field is padded with spaces (ASCII code 32)

Bits 4-8: Should contain the value 0

Output

AL = 0: The filename passed contains no wildcards

AL = 1: The filename passed contains wildcards

AL = 255: Invalid drive specifier

DS = Segment address of the first character after parsed filename

SI = Offset address of the first character after parsed filename

ES = FCB segment address

DI = FCB offset address

Remarks

The filename must end with an end character (ASCII code 0).

If the filename contains the wildcard "*", all corresponding fields in the FCB fill with the wildcard "?".

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 2AH**DOS (Version 1 and above)**

Get system date

Reads the current system date.

Input

AH = 2AH

Output AL = Day of the week (0=Sunday, 1=Monday, etc.)

 CX = Year

 DH = Month

 DL = Day

Remarks

DOS calls the clock driver to read the date.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 2BH**DOS (Version 1 and above)**

Set system date

Sets the current system date as returned by function 2AH (see above).

Input AH = 2BH

 CX = Year

 DH = Month

 DL = Day

Output AL = 0: O.K.

 AL = 255: Date incorrect

Remarks

The date passes to the clock driver.

If the PC does not have a realtime clock, the date remains in effect until the PC is switched off or rebooted.

If the date entry is incorrect, the PC retains the old date.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 2CH**DOS (Version 1 and above)**

Get system time

Gets the current system time.

Input AH = 2CH

Output CH = Hours

 CL = Minutes

 DH = Seconds

 DL = Hundredths of a second

Remarks

DOS calls the clock driver to read the time.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 2DH**DOS (Version 1 and above)**

Set system time

Sets the current system time.

Input

AH = 2DH

CH = Hours

CL = Minutes

DH = Seconds

DL = hundredths of a second

Output

AL = 0: O.K.

AL = 255: Incorrect time

Remarks

The time passes to the clock driver.

If the PC does not have a realtime clock, the time remains in effect until the PC is switched off or rebooted.

If the time entry is incorrect, the PC retains the old time.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 2EH**DOS (Version 1 and above)**

Set verify flag

Sets the verify flag. This flag determines whether data should be verified after a write operation to a block driver for proper transmission.

Input

AH = 2EH

DL = 0

AL = 0: Don't verify data

AL = 1: Verify data

Output

No output

Remarks

This flag can be controlled at the user level with the VERIFY ON and VERIFY OFF commands.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 2FH	DOS (Version 1 and above)
Get DTA address	

Returns the address of the DTA (Data Transmission Area), which serves as a data buffer for all FCB supported file accesses.

Input AH = 2FH

Output ES = DTA segment address
 BX = DTA offset address

Remarks

This function determines the address of the DTA, but not the DTA's size.

After the start of a program, the DTA starts at memory location 128 of the PSP and has a length of 128 bytes.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 30H	DOS (Version 2 and above)
Get MS-DOS version number	

Returns the DOS version number.

Input AH = 30H

Output AL = Major version number (e.g., version 2.01=2)
 AH = Minor version number (e.g., version 3.01=01)

Remarks

The major (whole) version number represents the number preceding the decimal point. For example, the version number 3.3 returns the major version number 3.

The minor (fractional) version number represents the number following the decimal point. It is always given as two digits. For example, Version 2.1 returns the minor version number 10 (0AH).

If the AL register contains a value of 0, the program runs under DOS Version 1. DOS Version 1.0 cannot use this function.

The contents of the DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 31H	DOS (Version 2 and above)
Terminate and stay resident	

Terminates the currently executing program and returns control to the calling program. The current program remains in memory for later recall.

Input AH = 31H
 AL = Return code
 DX = Number of paragraphs to be reserved

Output No output

Remarks

The return code in the AL register indicates whether the program called by it correctly executes. The calling program can read this number by calling function 77 (4DH). This value can be tested from within a batch file using the ERRORLEVEL and IF commands.

The number of 16-byte paragraphs to be reserved indicates how many bytes, beginning with the PSP, cannot be released for other uses.

Memory blocks reserved by function 48H are not affected by the value in the DX register because they can only be released by calling function 49H.

Interrupt 21H, Function 32H

DOS (Version 1 and above)

Get DPB pointer for any drive

Gets the pointer to a DOS Parameter Block for any disk drive.

Input AH = 32H

DL = Drive code (0=current, 1=A, 2=B, etc.)

Output Carry flag=1: Error

Carry flag=0: O.K.

DS:BX = FAR pointer to the DPB

Remarks

If the device indicated is a diskette drive, DOS accesses the drive to fill the DPB with data about the drive and the format of the diskette. For hard drives this information is already stored in the memory and no hardware access is required.

An error during the function call can only occur if an invalid drive code was indicated. If this occurs, error code 15 is returned in the AL register.

The construction of the DOS Parameter Block varies between DOS versions.

The contents of the AH, CX, DX, SI, DI, BP, CS, SS and ES registers are not affected by this function.

Interrupt 2FH, Subfunction 1

DOS (Version 3.0 and above)

Send file to print spooler

Reads the **Ctrl Break** flag. This determines whether DOS should test for active **Ctrl C** or **Ctrl Break** keys on each function call, or on character input/output calls. **Ctrl C** and **Ctrl Break** trigger interrupt 23H.

Input AH = 33H

AL = 0

Output DL = 0: Test only during character input/output

DL = 1: Test on every function call

Remarks

Since the **Ctrl Break** flag is not part of the environment block of a program, it affects all programs which call the DOS character functions that test for **Ctrl C** or the **Break** key.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 33H, Subfunction 1	DOS (Version 2.0 and above)
Set Ctrl Break flag	

Sets and unsets the **Ctrl Break** flag. This determines whether DOS should test for the activation of the **Ctrl C** or **Ctrl Break** keys on each DOS function call or character input/output calls. **Ctrl C** and **Ctrl Break** trigger interrupt 23H.

Input

AH = 33H

AL = 1

DL = 0: Test only during character input/output

DL = 1: Test on every function call

Output No output

Remarks

Since the **Ctrl Break** flag is not part of the environment block of a program, it affects all programs which call the DOS character functions that test for **Ctrl C** or the **Break** key.

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 34H	DOS (Version 2 and above)
Get INDOS flag pointer	

Returns the address of the INDOS flag. This is especially important for minimizing re-entry problems in TSR programs.

Input AH = 34H

Output ES:BX = FAR pointer to the INDOS flag

Remarks

The INDOS flag is a byte, which counts the recursive calls made to interrupt 21H. It contains the value 0 as long as DOS is not active. This means that DOS functions can be called from a TSR program. If this flag contains a value larger than 0, DOS is presently active, where a value greater than 1 indicates a corresponding number of recursive calls.

If this occurs, DOS cannot be interrupted with a function call from a TSR program, because in the worst case a system crash occurs.

See Chapter 35 for more information on TSRs.

The contents of the CX, DX, SI, DI, BP, CS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 35H**DOS (Version 2 and above)**

Get interrupt vector

Returns the current contents of an interrupt vector and the address of the interrupt routine that belongs to it.

Input

AH = 35H

AL = Interrupt number

Output

ES = Interrupt routine segment address

BX = Interrupt routine offset address

Remarks

To ensure compatibility with future versions of DOS, instead of reading the vector's contents directly from the interrupt vector table, call this function for reading an interrupt vector.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 36H**DOS (Version 2 and above)**

Get free disk space

Returns information about the device (the block driver) from which the available memory space can be calculated.

Input

AH = 36H

DL = Device code

Output

AX = 65535: Device unavailable

AX< 65535: Number of sectors per cluster

BX = Number of available clusters

CX = Number of bytes per sector

DX = Total number of clusters on the device

Remarks

This function identifies drive A as code 0, drive B as code 1, etc.

The remaining memory on the medium can be computed from the number of bytes per sector multiplied by the number of sectors per cluster, multiplied by the number of free clusters.

The contents of the SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 37H, Subfunction 00H**DOS (Version 2 and above)**

Get code for command line switch

Gets the command line switch character used in program calls.

Input	AH = 37H
	AL = 00H
Output	DL = ASCII character code

Remarks

The default command line switch character is the division character (/). For this reason few programs check this code, and will not react to changes made to this code.

The contents of the BX, CX, DH, SI, DI, BP and segment registers are not affected by this function.

Interrupt 21H, Function 37H, Subfunction 01H	DOS (Version 2 and above)
Set code for command line switch	

Sets the command line switch character used in program calls.

Input	AH = 37H
	AL = 01H
	DL = ASCII character code
Output	Carry flag=0: Character accepted
	Carry flag=1: Error

Remarks

The equal sign (=), slash (/) and hyphen (-) characters are valid command line switch characters. Other characters are invalid and are rejected.

Since few programs make use of the subfunction 00H of this function and instead always assume the (/) character as code for the command-line-switch, the call of subfunction 01H has effect on few programs. This includes in every case the various internal and external DOS programs which in this regard behave admirably.

The contents of the BX, CX, DX, SI, DI, BP and segment registers are not affected by this function.

Interrupt 21H, Function 38H	DOS (Version 2 and above)
Get country	

Determines country-specific parameters, which are set in the CONFIG.SYS file using the DOS COUNTRY command.

Input	AH = 38H
	AL = 0
	DS = Buffer segment address
	DX = Buffer offset address

Output No output

Remarks

Before the function call, function 30H should be used to determine the DOS version. This can help the programmer compensate for differences between DOS versions during the call and return of this function.

The buffer must have at least 32 bytes allocated for recording the various country-specific parameters.

Following the function call, the individual bytes of this buffer contain the following information:

Bytes 0-1: Date format

0=USA: Month-day-year

1=Europe: day-month-year

2=Japan: Year-month-day

Byte 2: ASCII code of the currency symbol

Byte 3: 0

Byte 4: ASCII code of the thousand character (comma/period)

Byte 5: 0

Byte 6: ASCII code of decimal character (period/comma)

Byte 7: 0

Bytes 8-31: reserved

The contents of the processor registers and the flag registers are not affected by this function.

Interrupt 21H, Function 38H, Subfunction 00H

DOS (Version 3 and above)

Get country

Gets the country-specific parameters that are currently set.

Input

AH = 38H

DS = Buffer segment address

DX = Buffer offset address

AL = 0: read current country parameters

AL = 1-254: Country code parameters to be read

AL = 255: Country code parameters to be read placed in the BX register

Output

Carry flag=0: O.K.

Carry flag=1: Invalid country code

Remarks

Before the function call, function 30H should be used to determine the DOS version. This can help the programmer compensate for differences between DOS versions during the call and return of this function.

The buffer must have at least 32 bytes allocated for recording the various country specific parameters.

Following the function call, the individual bytes of this buffer contain the following information:

Bytes 0-1: Date format

0=USA: Month-day-year

1=Europe: Day-month-year

2=Japan: Year-month-day

Bytes 2-6: Currency indicator (string terminated by an end character)

Byte 7: ASCII code of the thousand character (comma/period)

Byte 8: 0

Byte 9: ASCII code of decimal character (period/comma)

Byte 10: 0

Byte 11: ASCII code of the date separation character

Byte 12: 0

Byte 13: ASCII code of the time separation character

Byte 14: 0

Byte 15: Currency format

bit 0=0: Currency symbol before the value

bit 0=1: Currency symbol after the value

bit 1=0: No spaces between value and currency symbol

bit 1=1: Space between value and currency symbol

Byte 16: Precision (number of decimal places)

Byte 17: Time format

bit 0=0: 12-hour clock

bit 0=1: 24-hour clock

Bytes 18-21: Address of character conversion routine (see below)

Bytes 22-33: reserved

Addresses 18 to 21 are the offset and segment addresses of a FAR procedure, which is used for accessing the country specific characters from the character set of the PC. The routine views the AL register's contents as the ASCII code of a lower case letter that should be converted to a capital letter. If a capital letter exists, it is retained in the AL register after the call. If the letter doesn't exist, the contents of the AL register remain unchanged. For example, the routine could be used to convert a lower case "a" into a capital "A".

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and the flag registers are not affected by this function.

Interrupt 21H, Function 38H, Subfunction 01H**DOS (Version 3 and above)**

Set country

Sets the current country-specific parameters. These parameters can be read using function 38H, subfunction 0. Previous versions of DOS required country-specific settings from the CONFIG.SYS file using the COUNTRY command. This function allows the user to set and change these parameters after booting.

Input

AH = 38H

DX = 65535

AL = 1-254: Number of the country

AL > 254: Look in BX for country number

BX = Number of the country (if AL > 254)

Output

Carry flag=0: O.K.

Carry flag=1: Invalid country code

Remarks

Before the function call, function 30H should be used to determine that this command exists.

This function only allows setting of the country code, for which DOS has preset parameters. These parameters cannot be changed from this function.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 39H**DOS (Version 2 and above)**

Create subdirectory

Creates a new subdirectory on the specified device.

Input

AH = 39H

DS = Subdirectory path segment address

DX = Subdirectory path offset address

Output

Carry flag=0: Subdirectory created

Carry flag=1: Error (AX = error code)

AX=3: Path not found

AX=5: Access denied

Remarks

The subdirectory path passed is an ASCII string which is terminated by an end character (ASCII code 0).

If the subdirectory path contains a drive specifier, the indicated device is accessed. Otherwise DOS creates the subdirectory on the current device.

An error can occur if any element of the path designation doesn't exist, a subdirectory already exists by that name, or the directory to be made is a subdirectory of the root directory and it is already filled.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 3AH**DOS (Version 2 and above)**

Delete subdirectory

Deletes a subdirectory from the specified drive.

Input

AH = 3AH

DS = Subdirectory path segment address

DX = Subdirectory path offset address

Output

Carry flag=0: Subdirectory deleted

Carry flag=1: Error (AX = error code)

AX=3: Path not found

AX=5: Access denied

AX=6: Directory to be deleted is the current directory

Remarks

The subdirectory path passed is an ASCII string which is terminated by an end character (ASCII code 0). If the subdirectory path contains a drive specifier, the indicated device is accessed. Otherwise DOS deletes the subdirectory from the current device.

An error can occur if any element of the path designation doesn't exist, the subdirectory is the current directory, or the directory to be deleted still contains files.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 3BH**DOS (Version 2 and above)**

Set current directory

Sets the current subdirectory for the device indicated.

Input

AH = 3BH

DS = Subdirectory path segment address

DX = Subdirectory path offset address

Output

Carry flag=0: Subdirectory set

Carry flag=1: Error (AX = error code)

AX=3: Path not found

Remarks

The subdirectory path passed is an ASCII string which is terminated by an end character (ASCII code 0).

If the subdirectory path contains a drive specifier, the indicated device is accessed. Otherwise DOS deletes the subdirectory from the current device.

An error can occur if any element of the path designation doesn't exist.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 3CH	DOS (Version 2 and above)
Create or truncate file (handle)	

Creates a new file, or dumps the contents of an existing file (file size=0 bytes). This function call allows other functions to read or write to the open file.

Input

AH = 3CH

CX = File attribute

Bit 0=1: File is read only

Bit 1=1: Hidden file

Bit 2=1: System file

DS = Filename segment address

DX = Filename offset address

Output

Carry flag=0: O.K. (AX = file handle)

Carry flag=1: Error (AX = error code)

AX=3: Path not found

AX=4: No available handle

AX=5: Access denied

Remarks

The various bits of the file attribute can be combined with each other.

The filename must be available as an ASCII string terminated by an end character (ASCII code 0). The filename parameter can contain a driver specifier, path, filename and extension. No wildcards are allowed. If you omit the drive specifier or path, DOS accesses the current drive or current directory.

An error can occur if any element of the path designation doesn't exist, if the file must be created in the root directory which is already full, or if a file with the same name already exists but cannot be cleared because it is write protected (bit 0 in the file attribute byte = 1).

If the function call executed successfully, all other handle functions can be called with this handle once the file opens.

The file pointer is set to the first byte of the file.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 3DH	DOS (Version 2 and above)
Open file (handle)	

Opens an existing file for access by other functions.

Input

AH = 3DH

AL = Access mode

Bits 0-2: Read/write access

000(b)=File is read only

001(b)=File can only be written

010(b)=File can be read and written

Bit 3: 0(b)

Bits 4-6: File sharing mode

000(b)=Only current program can access the file (FCB mode)

001(b)=Only the current program can access the file

010(b)=Another program can read but not write the file

011(b)=Another program can write but not read the file

100(b)=Another program can read and write the file

Bit 7: Handle flag

0=Child program of the current program can access file handle

1=Current program can access file handle only

DS = Filename segment address

DX = Filename offset address

Output

Carry flag=0: O.K. (AX = file handle)

Carry flag=1: Error (AX = error code)

AX=1:Missing file sharing software

AX=2: File not found

AX=3: Path not found or file doesn't exist

AX=4: No handle available

AX=5: Access denied

AX=12: Access mode not permitted

Remarks

The filename must be available as an ASCII string terminated by an end character (ASCII code 0). The filename parameter can contain a driver specifier, path, filename and extension. No wildcards are allowed. If you omit the drive specifier or path, DOS accesses the current drive or current directory.

If the function call executes successfully, all other handle functions can be called with this handle once the file opens.

The file pointer is set to the first byte of the file.

DOS Version 2 uses only bits 0 to 2 of the access mode. All other bits, even under Version 3, should be 0 to ensure proper execution of the call.

DOS Version 3 uses the file sharing mode in bits 4 to 6 of the access mode only if the file is on a mass storage device which is part of a network. These three bits decide if and how the file, while it is open using the current call, may be accessed by other programs from other PCs on the network.

Error 12 can occur only under DOS Version 3 and only within a network when the file is already opened by another program and if no other program can gain access to that file.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 3EH**DOS (Version 2 and above)**

Close file (handle)

Writes any data in the DOS buffers to a currently open file, then closes the file. If changes occur to the file, the new file size and the last date and time of modification are added to the directory.

Input

AH = 3EH

BX = Handle to be closed

Output

Carry flag=0: O.K.

Carry flag=1: Error (AX = error code)

AX=6: Unauthorized handle or file not opened

Remarks

Do not accidentally call this function with the numbers of the previous handle (the numbers 0 to 4) because the standard input device or standard output device may close. This would leave you unable to enter characters from the keyboard or display characters on the screen.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 3FH**DOS (Version 2 and above)**

Read file or device (handle)

Reads a certain number of characters by using a handle from a previously opened file or device and passes the characters to a buffer. The read operation starts at the current file pointer position.

Input

AH = 3FH

BX = File or device handle

CX = Number of bytes to be read

DS = Buffer segment address

DX = Buffer offset address

Output Carry flag=0: O.K. (AX = number of bytes read)

Carry flag=1: Error (AX = error code)

AX=5: Access denied

AX=6: Illegal handle or file not open

Remarks

Characters can be read from a file or from a device (e.g., the standard input device [keyboard], which has the handle 0).

When the carry flag resets after the function call but the AX register has the value 0, this means that the file pointer has already reached the end of the file before the function call. So, no files could be read.

When the carry flag resets after the function call but the contents of the AX register are smaller than the contents of the CX register before the function call, this means that the desired number of bytes wasn't read because the end of the file was reached.

After the function call, the file pointer follows the last byte read.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 40H

DOS (Version 2 and above)

Write to file or device (handle)

Writes a certain number of characters from a buffer to an open file or device by using a handle. The write operation begins at the file pointer's current position.

Input AH = 40H

BX = File or device handle

CX = Number of bytes to be written

DS = Buffer segment address

DX = Buffer offset address

Output Carry flag=0: O.K. (AX = number of bytes written)

Carry flag=1: Error (AX = error code)

AX=5: Access denied

AX=6: Illegal handle or file not open

Remarks

Characters can be written to a file or to a device (e.g., the standard output device [screen], which has the handle 1).

When the carry flag resets after the function call but the AX register has the value 0, this means that the file pointer has already reached the end of the file before the function call. Therefore no files could be written.

When the carry flag resets after the function call but the contents of the AX register are smaller than the contents of the CX register before the function call, this means that the desired number of bytes were not written because the end of file was reached.

After the function call, the file pointer follows the last byte written.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 41H**DOS (Version 2 and above)**

Delete file (handle)

Deletes the filename passed to the function. Through the call of this function, a file is erased and its name is passed to the function.

Input

AH = 41H

DS = Filename segment address

DX = Filename offset address

Carry flag=0: O.K.

Carry flag=1: Error (AX = error code)

AX=2: File not found

AX=5: Access denied

Remarks

The filename must be available as an ASCII string terminated by an end character (ASCII code 0). The filename parameter can contain a drive specifier, path, filename and extension. No wildcards are allowed. If you omit the drive specifier or path, DOS accesses the current drive or current directory.

An error occurs when any element of the path designation doesn't exist or when the file has the attribute Read Only and therefore can not be written to or deleted. This attribute can be changed by using function 43H.

You cannot delete subdirectories or volume names with this function.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 42H**DOS (Version 2 and above)**

Move file pointer (handle)

Moves the file pointer of a previously opened file by using its handle. This allows random access because the individual records don't have to be read in sequence. The new file pointer position is given as an offset from the current position, either from the beginning of the file or from the end of the file. The offset itself is indicated as a 32-bit number.

Input

AH = 42H

AL = Offset code

AL=0: Offset is relative to the beginning of the file

AL=1: Offset is relative to the current position of the file pointer

AL=2: Offset is relative to the end of the file

BX = Handle
 CX = High word of the offset
 DX = Low word of the offset

Output Carry flag=0: O.K.

DX=High word of the file pointer
 AX=Low word of the file pointer
 Carry flag=1: Error (AX = error code)
 AX=1: Illegal offset code
 AX=6: Illegal handle or File not open

Remarks

If offset codes 1 and 2 are accessed, negative offsets may be used to move the file pointer backwards or to place the pointer at the beginning of the file. It's possible to set the file pointer before the end of the file, which causes an error during the next read or write access to the file.

The position of the file pointer passed after the function call is always relative to the beginning of the file. The offset code used during the function call is independent of this file pointer position.

Passing offset code 2 and offset 0 returns the size of the file. This action moves the file pointer to the last byte of the file and the pointer's position returns to the calling program after the function call.

The contents of the BX, CX, , SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 43H, Subfunction 0

DOS (Version 2 and above)

Get file attributes

Determines file attributes.

Input AH = 43H
 AL = 0
 DS = Filename segment address
 DX = Filename offset address

Output Carry flag = 0: O.K. (CX = file attribute)
 Bit 0=1: File can be read but not written
 Bit 1=1: File hidden (not displayed on DIR)
 Bit 2=1: File is a system file
 Bit 3=1: File is the volume name
 Bit 4=1: File is a subdirectory
 Bit 5=1: File was changed since the last date/time

Carry flag = 1: Error (AX = error code)

AX=1: Unknown function code

AX=2: File not found

AX=3: Path not found

Remarks

The filename must be available as an ASCII string terminated by an end character (ASCII code 0). The filename parameter can contain a driver specifier, path, filename and extension. No wildcards are allowed. If you omit the drive specifier or path, DOS accesses the current drive or current directory.

An error occurs when any element of the path designation or the file does not exist. The contents of the BX, CX, , SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 43H, Subfunction 1

DOS (Version 2 and above)

Set file attributes

Sets the file attributes.

Input

AH = 43H

AL = 1

CX = File attributes

Bit 0=1: File can be read but not written

Bit 1=1: File hidden (not displayed on DIR)

Bit 2=1: File is a system file

Bit 3=0

Bit 4=0

Bit 5=1: File was changed since the last date/time

DS = Filename segment address

DX = Filename offset address

Output

Carry flag=0: O.K.

Carry flag=1: Error (AX = error code)

AX=1: Unknown function code

AX=2: File not found

AX=3: Path not found

AX=5: Attribute cannot be changed

Remarks

The filename must be available as an ASCII string terminated by an end character (ASCII code 0). The filename parameter can contain a driver specifier, path, filename and extension. No wildcards are allowed. If you omit the drive specifier or path, DOS accesses the current drive or current directory.

An error occurs when any element of the path designation or the file does not exist.

Neither subdirectories nor volume names can be accessed with this function. For this reason bits 3 and 4 of the file attribute must be 0 during the function call. If you attempt to access a subdirectory or a volume name anyway, the function returns error code 5.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 1	DOS (Version 2.0 and above)
IOCTL: Set device information	

Permits access of a character driver's device attribute.

Input

AH = 44H

AL = 0

BX = Handle

Carry flag=0: O.K. (DX = device attribute)

Bit 14=1: Processes control characters through IOCTL

Bit 7=1: Character driver

Bit 5=0: Cooked mode operation

Bit 5=1: Raw mode operation

Bit 3=1: Clock driver operation

Bit 2=1: NUL driver operation

Bit 1=1: Console output driver (screen)

Bit 0=1: Console input driver (keyboard)

Carry flag=1: Error (AX = error code)

AX=1: Unknown function code

AX=6: Handle not opened or does not exist

Remarks

A handle is passed (not the name of the addressed character driver which must be connected with this driver). This can be one of the five pre-assigned handles (0 to 4). A handle could have been previously opened for a certain device with the help of the Open function (function 3DH), and then passed to the function. For example, since the standard input and output devices (handles 0 and 1) can be redirected, this method assures that the indicated device is accessed.

If bit 7 in the device attribute is unequal to 1, the driver addressed is not a character driver and the significance of the individual bits in the device attribute disagrees with those of the device driver. The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 1**DOS** (*Version 2.0 and above*)

IOCTL: Set device information

Sets the character device attributes.

Input

AH = 44H

AL = 1

BX = Handle

CX = Number of bytes written

DX = Device attributes

Bit 14=1: Processes control characters through IOCTL using subfunctions 2 and 3

Bit 7=1: Character driver

Bit 5=0: Cooked mode operation

Bit 5=1: Raw mode operation

Bit 3=1: Clock driver operation

Bit 2=1: NUL driver operation

Bit 1=1: Console output driver (screen)

Bit 0=1: Console input driver (keyboard)

Carry flag=0: O.K.

Output

Carry flag=1: Error (AX = Error code)

AX=1: Unknown function code

AX=6: handle not opened or handle does not exist

Remarks

A handle is passed but it is not the name of the addressed character device, which must be connected with this device. This can be one of the five pre-assigned handles (0 to 4). A handle could have previously been opened, with the Open function, for a certain device and then passed to the function. For example, since the standard input and output devices (handles 0 and 1) can be redirected, this method assures that the indicated device is accessed.

To change various device attribute bits with this function, use subfunction 0 to read the device attributes first. Then this subfunction can reset the device attribute bits in the device driver.

If bit 7 in the device attribute is unequal to 1, the driver addressed is not a character driver. The meanings of the individual bits in the device attribute disagree with those in the device driver.

This function is especially useful for switching between cooked mode and raw mode within a character driver (bit 5).

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 2	DOS (Version 2.0 and above)
IOCTL: Read data from character device	

Reads data from a character device. This function defines the number of bytes of data to read from the buffer, which contains the data taken from the character device.

Input	AH = 44H
	AL = 2
	BX = Handle
	CX = Number of bytes to be read
	DS = Buffer segment address
	DX = Buffer offset address
Output	Carry flag=0: O.K. (AX = Number of bytes sent)
	Carry flag=1: Error (AX = Error code)
	AX=1: Unknown function code
	AX=6: Handle not opened or does not exist

Remarks

A handle is passed, but it is not the name of the addressed character device which must be connected with this device. This can be one of the five pre-assigned handles (0 to 4). A handle could have previously been opened with the Open function (function number 3DH) for a certain device, then passed to the function. For example, since the standard input and output devices (handles 0 and 1) can be redirected, this method assures that the indicated device is accessed.

An error always occurs if the handle passed is connected with a block driver instead of a character driver.

The driver defines the data type and structure.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 3	DOS (Version 2.0 and above)
IOCTL: Send data to character device	

Sends data from an application program directly to a character device. The calling function defines the number of bytes to be transferred from a buffer to the device.

Input	AH = 44H
	AL = 3
	BX = Handle
	CX = Number of bytes to be transmitted
	DS = Buffer segment address
	DX = Buffer offset address

Output Carry flag=0: O.K.

AX=Number of bytes sent

Carry flag=1: Error (AX = Error code)

AX=1: Unknown function code

AX=6: Handle not opened or does not exist

Remarks

A handle is passed, but it is not the name of the addressed character device which must be connected with this device. This can be one of the five pre-assigned handles (0 to 4). A handle could have previously been opened with the Open function (function number 61) for a certain device, then passed to the function. For example, since the standard input and output devices (handles 0 and 1) can be redirected, this method assures that the indicated device is accessed.

An error always occurs if the handle passed is connected with a block driver instead of a character driver.

The driver defines the data type and structure.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 4

DOS (Version 2.0 and above)

IOCTL: Read data from block device

Reads data for an application directly from a block device. The calling function defines the number of bytes to be copied by the device into a buffer.

Input

AH = 44H

AL = 4

BX = Device designation

CX = Number of bytes to be read

DS = Buffer segment address

DX = Buffer offset address

Carry flag=0: O.K.

AX = Number of bytes sent

Output

Carry flag=1: Error (AX = Error code)

AX=1: Unknown function code

AX=15: Unknown device

Remarks

Instead of defining the device driver, the device designation parameter defines the device from which data will be received. Code 0 represents device A:, 1 represents device B:, etc.

The driver defines the data type and structure. The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 4	DOS (Version 6.0 and above)
IOCTL / DoubleSpace: Write internal caches	

Instructs DoubleSpace to write the contents of its internal cache buffers for cluster data, BitFAT and MDFAT to disk.

Input

AH = 44H

AL = 04H

BX = Device designation (0 = A)

CX = Number of bytes to be read

DS:DX = FAR pointers to buffer (see below)

Output

Carry flag = 0: OK

Carry flag = 1: Error

Remarks

In BX you can enter the device designation of any compressed drive.

The buffer referred to by DS:DX upon calling the function must contain the following:

Offset	Contents	TaskType	
+00H	'DM'	For identification of Microsoft DoubleSpace	1 word
+02H	'F'	For "Flush" instruction	1 byte
+03H	0	Receives error code following call	1 word
+05H	0, 0, 0, 0, 0	Five fill bytes	5 bytes

Length: 10 bytes

Following the function call, the word at offset address 03H of the buffer contains the character combination "OK" if the function was carried out successfully.

Interrupt 21H, Function 44H, Subfunction 4	DOS (Version 6.0 and above)
IOCTL / DoubleSpace: Write internal caches and invalidate	

Instructs DoubleSpace to write the contents of its internal cache buffers for cluster data, BitFAT and MDFAT to disk, then declare the contents of these buffers invalid.

Input

AH = 44H

AL = 04H

BX = Device designation (0 = A)

CX = Number of bytes to be read

DS:DX = FAR pointer to buffer (see below)

Carry flag = 0: OK

Carry flag = 1: Error

Remarks

In BX you can enter the device designation of any compressed drive.

The buffer referred to by DS:DX upon calling the function must contain the following:

Offset	Contents	TaskType	
+00H	'DM'	For identification of Microsoft DoubleSpace	1 word
+02H	'I'	For "Invalidate" instruction	1 byte
+03H	0	Receives error code following call	1 word
+05H	0, 0, 0, 0, 0	Five fill bytes	5 bytes

Length: 10 bytes

Following the function call, the word at offset address 03H of the buffer contains the character combination "OK" if the function was carried out successfully.

Interrupt 21H, Function 44H, Subfunction 5

DOS (Version 2.0 and above)

IOCTL: Send data to block device

Sends data from an application program directly to a character device. The calling function defines the number of bytes to be transferred from a buffer to the device.

Input

AH = 44H
 AL = 5
 BX = Device designation
 CX = Number of bytes to be sent
 DS = Buffer segment address
 DX = Buffer offset address

Output

Carry flag=0: O.K.
 AX=Number of bytes sent
 Carry flag=1: Error (AX = Error code)
 AX=1: Unknown function code
 AX=15: Unknown device

Remarks

Instead of defining the device driver, the device designation parameter defines the device from which data will be received. Code 0 represents device A:, 1 represents device B:, etc.

The driver defines the data type and structure. The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 6	DOS (Version 2.0 and above)
IOCTL: Read input status	

Determines whether a device driver can transmit data to an application program.

Input	AH = 44H AL = 6 BX = Handle
Output	Carry flag=0: O.K. (AX = Input status) AX=0: Driver not ready AX=255: Driver ready Carry flag=1: Error (AX = Error code) AX=1: Unknown function code AX=5: Access denied

Remarks

The handle passed can refer to either a character driver or a file.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 7	DOS (Version 2.0 and above)
IOCTL: Read output status	

Determines whether a device driver can receive data from an application program.

Input	AH = 44H AL = 7 BX = Handle
Output	Carry flag=0: O.K. (AX = Output status) AX=0: Driver not ready AX=255: Driver ready Carry flag=1: Error (AX = Error code) AX=1: Invalid function number AX=5: Access denied

Remarks

The handle passed can refer to either a character driver or a file.

If the handle refers to a file, the block device driver signals its readiness to receive data, even if the medium containing the file is full and no additional data can be appended to the end of the file.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 8	DOS (Version 3.0 and above)
IOCTL: Test for changeable block device	

Determines whether the block device medium (e.g., disk, hard drive, etc.) can be changed.

Input	AH = 44H
	AL = 8
	BL = Device designation
Output	Carry flag=0: O.K. (AX=status code)
	AX=0: Medium changeable
	AX=1: Medium unchangeable
	Carry flag=1: Error (AX = Error code)
	AX=1: Invalid function number
	AX=15: Invalid drive number

Remarks

The device designation parameter defines the device being addressed instead of the device driver. Code 0 represents device A:, 1 represents device B:, etc.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 9	DOS (Version 3.1 and above)
IOCTL: Test for local or remote drive	

Determines whether a drive (block device) is local (part of the PC making the inquiry) or remote (part of another PC in a network).

Input	AH = 44H
	AL = 9
	BL = Device designation
Output	Carry flag=0: O.K.
	DX = device attribute
	Bit 12=0: Local
	Bit 12=1: Remote
	Carry flag=1: Error (AX = Error code)

AX=1: Invalid function number

AX=15: Invalid drive specification

Remarks

You can access this subfunction only if networking software has previously been installed.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 0AH	DOS (Version 3.1 and above)
IOCTL: Test for local or remote handle	

Determines whether a file associated with this handle is local (part of the PC making the inquiry) or remote (part of another PC in a network).

Input

AH = 44H

AL = 0AH

BX = Handle

DX = IOCTL code

Bit 15 = 0: Local

Bit 15 = 1: Remote

Output

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function number

AX=6: Handle not opened or does not exist

Remarks

You can access this subfunction only if networking software has been installed.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 0BH	DOS (Version 3.0 and above)
IOCTL: Change retry count	

Sets the variables that specify the number of attempts at file access. One PC within a network may try to access a file that is already being accessed by another PC. The PC attempting access repeats the file access procedure the number of times and the number of waiting periods defined by these variables.

Input

AH = 44H

AL = 0BH

BX = Number of attempts

CX = Waiting time between attempts

Output Carry flag=0: O.K.
 Carry flag=1: Error (AX = Error code)
 AX=1: Invalid function number

Remarks

You can only access this subfunction if networking software has previously been installed.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 0CH	DOS (Version 3.3 and above)
IOCTL: Character driver access	

Permits character driver access to code pages. Since code pages are only supported by certain character drivers, these functions are available to these drivers only.

Input AH = 44H
 AL = 0CH
 BX = Handle
 CH = Device designation
 CL = Device subfunction (see below)
 DS:DX = FAR pointer to buffer and other information

Output Carry flag=0: O.K.
 Carry flag=1: Error (AX = Error code)

Remarks

The CH register specifies the device designation, using one of the following numbers:

- 1 = Serial device (COM1, COM2, COM3 or COM4)
- 2 = Display device (CON)
- 5 = Parallel device (LPT1, LPT2 or LPT3)
- 0 = None of the above

The following subfunctions can be addressed through the CL register:

- 45H = Set iteration count for printer driver (DOS 3.3 and up)
- 4AH = Select code page (DOS 3.3 and up)
- 4CH = Start code page initialization (DOS 3.3 and up)
- 4DH = End code page initialization (DOS 3.3 and up)
- 5FH = Set display mode (DOS 4.0 and up)
- 65H = Get iteration count for printer driver (DOS 3.3 and up)

6AH = Query current code page (DOS 3.3 and up)

6BH = Query code page initialization list (DOS 3.3 and up)

7FH = Get display mode (DOS 4.0 and up)

Calling this function requires the following steps:

1. Call function 60H to query the current parameter. Verify and store the current parameter for later recall.
2. Call function 40H to set the new parameter.
3. Execute function 44H, subfunction 0CH.
4. Call function 40H to restore the original parameter.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 0DH
IOCTL: Block driver access

DOS (Version 3.2 and above)

Permits block driver access to code pages. Since code pages are only supported by certain character drivers, these functions are available to these drivers only.

Input

AH = 44H

AL = 0DH

BL = Device designation

CH = 08H

CL = Device subfunction (see below)

DS:DX = FAR pointer to buffer and other information

Output

Carry flag=0: O.K.

Carry flag=1: Error (AX = Error code)

Remarks

The device designation parameter defines the device being addressed instead of the device driver. Code 0 represents device A:, 1 represents B:, etc.

The following subfunctions can be addressed through the CL register:

40H = Set device parameters (DOS 3.2 and up)

41H = Write track on logical drive (DOS 3.2 and up)

42H = Format track on logical drive (DOS 3.2 and up)

46H = Set media ID (DOS 4.0 and up)

60H = Determine device parameters (DOS 3.2 and up)

61H = Read track on logical drive (DOS 3.2 and up)

62H = Verify track on logical drive (DOS 3.2 and up)

66H = Get media ID (DOS 3.2 and up)

68H = Sense media type (DOS 5.0 and up)

Calling this function requires the following steps:

1. Call function 60H to query the current parameter. Verify and store the current parameter for later recall.
2. Call function 40H to set the new parameter.
3. Execute function 44H, subfunction 0CH.
4. Call function 40H to restore the original parameter.
5. The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 0EH	DOS (Version 3.2 and above)
IOCTL: Get logical drive map	

Determines whether multiple logical drives are assigned to one block device.

Input

AH = 44H

AL = 0EH

BL = Device code (0=current, 1=A, 2=B, etc.)

Output

Carry flag=0: O.K.

AL = 0 : Device is a logical device

AL =>1 : Number of logical device (1=A:, 2=B:, etc.)

Carry flag=1: Error (AX = error code)

Remarks

This function is of interest mainly in relation to system with only one diskette drive, because this drive can be addressed as A and B. With the help of this function it can be determined if the drive was last addressed as either A or B. If the drive is addressed through this query, a DOS message for insertion of a diskette may be avoided during the access to the drive.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 0FH	DOS (Version 3.2 and above)
IOCTL: Set logical drive map	

With this function the logical drive designation can be determined, through which a device is addressed during the next access.

Input

AH = 44H

AL = 0FH

BL = Device code (0=current, 1=A, 2=B, etc.)

Output Carry-Flag=0: o.k., in this case

AL = Device code, which the drive is addressed during the next access.

Carry-Flag=1: Error, in this case AX = Error-Code

Remarks

This function is interesting mainly in relation to systems with only one diskette drive, because this drive can be addressed as either A or B. In this case, the diskette drive can be switched with the help of this function from A to B. Since the function works alternately, a subsequent call will switch a drive to A again.

The contents of the BX, CX, DX, SI, DI, BP and the segment Registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 10H	DOS (Version 5.0 and above)
IOCTL: Query IOCTL handle	

With this function the availability of an IOCTL-function, which is not available in version 3.2, can be queried. This function is only available after version 5.0.

Input

AH = 44H

AL = 10H

BX = Handle, whose device should be considered during the call of the IOCTL function.

CL = Number of the desired IOCTL function

Output

Carry-Flag=0: Function is available in relation to the indicated Handle

Carry-Flag=1: Error, function not available in relation to the Handle

AX = 1 (Function not available)

Remarks

While the availability of a certain IOCTL function can be tested with the help of this function only in relation to the device which hides behind a Handle, the IOCTL function 11H makes a test in relation to a certain device possible.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 44H, Subfunction 11H	DOS (Version 5.0 and above)
IOCTL: IOCTL device	

After version 5.0, this function permits the query of the availability of this IOCTL function, before the call of an IOCTL function which was not included in the functions of Version 3.2.

Input

AH = 44H

AL = 11H

BX = Device-Number (0=current, 1=A, 2=B, etc.)

CL = Number of the desired IOCTL function

Output Carry-Flag=0: Function is available in relation to the indicated device

 Carry-Flag=1: Error, function is not available in relation to the device

 AX = 1 (Function not available)

Remarks

While the availability of a certain IOCTL function can be tested with the help of this function only in relation to the device which hides behind a Handle, the IOCTL function 10H makes a test possible in relation to a certain Handle and the device which is hidden behind it.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 45H**DOS (Version 2.0 and above)**

Duplicate handle

Creates a duplicate of the handle passed. This duplicate handle interfaces with the same file or device as the first handle. If the first handle refers to a file, the value of the first handler's file pointer joins with the file pointer of the duplicate handle.

Input AH = 45H

 BX = Handle

Output Carry flag=0: O.K. (AX = New handle)

 Carry flag=1: Error (AX = Error code)

 AX=4: No additional handle available

 AX=6: Handle not opened or does not exist

Remarks

Without having to close the file, this function updates a file directory entry after its modification. A file can be closed using function 62 (3EH).

If the file pointer of one of the two handles changes position due to the call of a read or write function, the other file pointer also changes automatically.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 45H**DOS (Version 2.0 and above)**

IOCTL: Duplicate handle

Refers a second file handle to the save device or file as the first file handle. The second handle's file pointer also contains the same value as the first handle's file pointer.

Input AH = 46H

 BX = First handle

 CX = Second handle

Output Carry flag=0: O.K.

Carry flag=1: Error (AX = Error code)

AX=4: No additional handle available

AX=6: Handle not opened or does not exist

Remarks

If the function call connects the second handle to an open file, the file closes before the forced duplication.

If the file pointer of one of the handles changes position due to the call of a read or write function, the other file pointer also changes automatically.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 47H

DOS (Version 2.0 and above)

Get current directory

Gets an ASCII string listing the complete path designation of the current directory of the indicated device. This string passes to the specified buffer.

Input

AH = 47H

DL = Device designation

DS = Buffer segment address

SI = Buffer offset address

Output

Carry flag=0: O.K.

Carry flag=1: Error (AX=Error code)

AX=15: Invalid drive specification

Remarks

The device designation parameter defines the device being addressed instead of the device driver. Code 0 represents the current device, 1 represents device A:, etc.

The path description in the buffer terminates with an end character (ASCII code 0). This description has no drive specifier or \ character (root directory specifier). If the root directory is the current directory, the end character becomes the first character in the buffer.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 48H

DOS (Version 2.0 and above)

Allocate memory

Reserves an area of memory for program use.

Input

AH = 48H

BX = Number of paragraphs to be reserved

Output Carry flag=0: O.K.

 AX=Memory area segment address)

 Carry flag=1: Error (AX = Error code)

 AX=7: Memory control block destroyed

 AX=8: Insufficient memory

 BX = Number of paragraphs available

Remarks

A paragraph consists of 16 bytes.

If memory allocation was successfully executed, the allocated range begins at address AX:0000.

This function always fails when executed from within a COM program because the PC assigns the total amount of free memory to a COM program when it executes.

The contents of the CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 49H**DOS (Version 2.0 and above)**

Release memory

Releases memory previously allocated by function 72 (49H-see above) for any purpose.

Input AH = 49H

 ES = Memory area segment address

Output Carry flag=0: O.K.

 Carry flag=1: Error (AX = Error code)

 AX=7: Memory control block destroyed

 AX=9: Incorrect memory area passed in ES

Remarks

Since DOS knows the size of the memory area to be released, no parameter exists for passing memory size.

If the wrong segment address appears in the ES register during the function call, memory assigned to another program can be released. This can lead to a system crash or other consequences.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 4AH**DOS (Version 2.0 and above)**

Modify memory allocation

Changes the size of a memory area previously reserved using function 72 (3FH-see above).

Input

AH = 4AH

BX = New memory area size in paragraphs

ES = Memory area segment address

Output

Carry flag=0: O.K.

Carry flag=1: Error (AX = Error code)

AX=7: Memory control block destroyed

AX=8: Insufficient memory

BX = Number of paragraphs available

Remarks

A paragraph has 16 bytes.

If the wrong segment address appears in the ES register during the function call, memory assigned to another program can be released. This can lead to a system crash or other consequences.

Since the PC assigns the total amount of free memory to a COM program when it executes, this function call always fails when executed from within a COM program.

COM programs should use this function to release all unnecessary memory since all RAM becomes part of a COM program. This is especially important before calling the EXEC function (function number 75 (4BH)).

The contents of the CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 4BH, Subfunction 0**DOS (Version 2.0 and above)****Execute program**

Executes another program from within a program and continues execution of the original program after the called program finishes its run. The function requires the name of the program to be executed and the address of a parameter block, which contains information that is important to the function.

Input

AH = 4BH

AL = 0

ES = Parameter block segment address

BX = Parameter block offset address

DS = Program name segment address

DX = Program name offset address

Output

Carry flag=0: O.K.

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function number

AX=2: Path or program not found

AX=5: Access denied

AX=8: Insufficient memory

AX=10: Wrong environment block

AX=11: Incorrect format

Remarks

The directory name passed is an ASCII string which is terminated by an end character (ASCII code 0). It can contain a path designation and drive specifier. No wildcards are allowed. If no drive specifier or path designation exists, the function accesses the current drive or directory.

Only EXE or COM programs can be executed. To execute a batch file, the command processor (COMMAND.COM) must be called using the /c parameter followed by the name of the batch file.

The parameter block must have the following format:

Bytes 0-1:	Environment block segment address
Bytes 2-3:	Command parameter offset address
Bytes 4-5:	Command parameter segment address
Bytes 6-7:	First FCB offset address
Bytes 8-9:	First FCB segment address
Bytes 10-11:	Second FCB offset address
Bytes 12-13:	Second FCB segment address

If the segment address of the environment block is a 0, the called program has the same environment block as the calling program.

The command parameters must be stored so that the parameter string begins with a byte representing the number of characters in the command line. Next follow the individual ASCII characters, which are terminated by a carriage return (ASCII code 13) (this carriage return is not counted as a character).

The first FCB passed is copied to the PSP of the called program starting at address 5CH. The second FCB passed is copied to the PSP of the called program starting at address 6CH. If the called program does not obtain information from the two FCBs, any desired value can be entered into the FCB fields at the parameter block.

After the call of this function, all registers are destroyed except the CS and IP registers. For later recall, save their contents before the function call.

The program called should have all the handles available to the calling program.

Interrupt 21H, Function 4BH, Subfunction 3

DOS (Version 2.0 and above)

Execute overlay

Loads a second program into memory as an overlay without automatically executing the second program.

Input

AH = 4BH

AL = 3

ES = Parameter block segment address

BX = Parameter block offset address
 DS = Program name segment address
 DX = Program name offset address

Output Carry flag=0: O.K.

Carry flag=1: Error (AX = Error code)
 AX=1: Invalid function number
 AX=2: Path or program not found
 AX=5: Access denied
 AX=8: Insufficient memory
 AX=10: Wrong environment block
 AX=11: Incorrect format

Remarks

The directory name passed is an ASCII string which is terminated by an end character (ASCII code 0). It can contain a path designation and drive specifier. No wildcards are allowed. If no drive specifier or path designation exists, the function accesses the current drive or directory.

Only EXE or COM programs can be executed. To execute a batch file, the command processor (COMMAND.COM) must be called using the /c parameter followed by the name of the batch file.

The parameter block must have the following format:

Byte 0-1: Segment address where the overlay will be stored
 (offset address=0)

Byte 2-3: Relocation factor

The relocation factor requires the value 0 for COM programs. Use the segment address at which the program should load when accessing EXE programs. The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 4BH, Subfunction 05H	DOS (Version 5.0 and above)
Set EXECSTATE	

Applications which load other programs or overlays without the DOS Exec function, must use this function after version 5.0 of DOS to avoid problems during loading of the programs and overlays.

Input AH = 4BH
 AL = 05H
 DS:DX = FAR-pointer to the Exec-State structure

Output Carry-Flag=0: o.k.

Carry-Flag=1: Error, in this case

AX = Error-Code

- 1: unknown function code
- 2: program not found
- 3: program not found
- 4: too many files opened
- 5: access denied
- 8: insufficient storage area
- 10: wrong Environment-Block
- 11: wrong format

Remarks

The call of this function must occur between the loading of the program or overlay and its execution. Between the call of this function and the start of the program or overlay, neither DOS- nor BIOS functions, nor any other software interrupts may be called.

In the ExecState structure, information about the overlay or program is stored. It consists of 18 Bytes and must have the following format.

Format of the ExecState structure		
Addr.	Content	Type
+00H	Reserved, must contain 0	1 WORD
+02H	1 = EXE-program 2 = Overlay	1 WORD
+04H	Pointer to an ASCIIZ string with the name of the program or overlay (Path not allowed in this string)	1 PTR
+08H	Segment address of the PSP of the of the program or overlay	1 WORD
+0AH	Jump location to the program or overlay	1 PTR
+0EH	Program- or overlay size including PSP	1 DWORD
Length: 12H (18 Bytes)		

Interrupt 21H, Function 4CH	DOS (Version 2.0 and above)
Terminate with return code	

Terminates a program and passes an end code for which function 77 (4DH-see below) searches. This function releases the memory previously occupied by the terminated program.

Input AH = 4CH

AL = Return code

Output No output

Remarks

This function may be used for program termination instead of the other functions listed earlier.

This function call restores the contents of the three interrupt vectors that were stored in the PSP when the program started execution.

Before passing control to the calling program, all handles opened by this program close, along with the corresponding files. This is not applicable to files accessed using FCBs.

A batch file can test for the return code using the ERRORLEVEL and IF batch commands.

Interrupt 21H, Function 4DH	DOS (Version 2.0 and above)
Get return code	

Checks a program, called from another program by the EXEC function, for the return code passed by the called program when it terminates.

Input	AH = 4DH
Output	AH = Type of program termination
	AH=0: Normal end
	AH=1: End through Ctrl C or Break
	AH=2: Device access error
	AH=3: Call of function 49 (31H)
	AL = Return code

Remarks

This function reads the return code of the called program only once.

The contents of the AX, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and flag registers are not affected by this function. The contents of all other registers may change.

Interrupt 21H, Function 4EH	DOS (Version 2.0 and above)
Search for first match	

Searches for the first occurrence of the filename listed. The file can have certain attributes, so a search can be made through subdirectories and volume names.

Input	AH = 4EH
CX =	File attribute
DS =	Filename segment address
DX =	Filename offset address

Output Carry flag=0: O.K.

Carry flag=1: Error (AX = Error code)

AX=2: Path not found

AX=18: No file with the attribute found

Remarks

The directory name passed is an ASCII string which is terminated by an end character (ASCII code 0). It can contain a path designation and drive specifier. No wildcards are allowed. If no drive specifier or path designation exists, the function accesses the current drive or directory.

The search defaults to normal files (attribute 0). Any set attribute bits extends the search to normal files and any other file types.

If a matching file occurs, the first 43 bytes of the DTA contain the following information about this file:

Bytes 0-20:	Reserved
Byte 21:	File attribute
Bytes 22-23:	Time of last modification to file
Bytes 24-25:	Date of last modification to file
Bytes 26-27:	Low word of file size
Bytes 28-29:	High word of file size
Bytes 30-42:	ASCII filename and extension terminated by an end character (ASCII code 0)

This function may only be called to search for the first occurrence of a file. If you want to search for a group of files using wildcards, function 4FH (see below) must be called.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 4FH

DOS (Version 2.0 and above)

Search for next match (handle)

Searches for subsequent occurrences of the filename listed after function 78 (above) executed successfully.

Input AH = 4FH

Output Carry flag=0: O.K.

Carry flag=1: Error (AX=Error code)

AX=18: No other files found with this attribute

Remarks

If a matching file occurs, the first 43 bytes of the DTA contain the following information about this file:

Bytes 0-20:	Reserved
-------------	----------

Byte 21: File attribute

Bytes 22-23: Time of last modification to file

Bytes 24-25: Date of last modification to file

Bytes 26-27: Low word of file size

Bytes 28-29: High word of file size

Bytes 30-42: ASCII filename and extension terminated by an end character (ASCII code 0)

This function can only be called if function 4EH has been called once and if the DTA remains unchanged.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21(h), function 50(h) DOS

Reserved (Version 2 and up)

Interrupt 21(h), function 51(h) DOS

Reserved (Version 2 and up)

Interrupt 21(h), function 52(h) DOS

Reserved (Version 2 and up)

Interrupt 21(h), function 53(h) DOS

Reserved (Version 2 and up)

Interrupt 21H, Function 50H

DOS (Version 2.0 and above)

Set active PSP

This function sets the current PSP, as opposed to the DOS. It is required for example in TSR programs to obtain data during file access from you PSP and not from the PSP of the current foreground program.

Input

AH = 50H

BX = Segment address of the PSP

Output

Carry-Flag=0: o.k.

Carry-Flag=1: Error, in this case

AX = Error-code

Remarks

DOS assumes that a PSP always starts at the Offset address 0000H in a segment. This should be noted during the call of this function.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 51H**DOS (Version 2.0 and above)**

Determine active PSP

With this function the Segment address of the PSP of the current program can be determined. This is mostly used by TSR programs which note the address of the PSP of the foreground program, before setting their own PSP with the help of the function 50H.

Input AH = 51H

Output Carry-Flag=0: o.k. in this case

BX = Segment address of the current PSP

Carry-Flag=1: Error, in this case

AX = Error-code

Remarks

The indicated PSP always begins at the Offset address 0000H relative to the indicated Segment address.

Function 62H should be preferred to the call of this function, which is specifically designed for this purpose and in contrast to function 51H is documented openly. However, it is only available after version 3.0 of DOS, while the function 51H was already introduced with version 2.0.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 53H**DOS (Version 2.0 and above)**

Convert BPB to DPB

This function returns a pointer to the DOS-Info-Block (DIB). With this function much interesting information can be obtained, which otherwise would not be accessible to a program.

Input AH = 52H

Output Carry-Flag=0: o.k. in this case

ES:BX =FAR-pointer to the DIB

Carry-Flag=1: Error, in this case

AX = Error-code

Remarks

The structure of the DOS-Info-Block can be found in Chapter 6.

The contents of the CX, DX, SI, DI, BP, CS, DS and SS registers are not affected by this function.

Interrupt 21H, Function 53H**DOS (Version 2.0 and above)**

Convert BPB to DPB

This undocumented function converts an available BIOS-Parameter-Block (BPB) to a Drive-Parameter-Block (DPB).

Input	AH = 53H
	DS:SI = FAR-pointer to the buffer with the BPB to be converted
	ES:BP = Pointer to the buffer, in which the Drive-Parameter-Block should be created (see below)
Output	Carry-Flag=0: o.k.
	Carry-Flag=1: Error, in this case

AX = Error-code

Remarks

A description of the structure of the BIOS and the Drive-Parameter-Block, can be found in Chapter 6.

Interrupt 21H, Function 54H	DOS (Version 2.0 and above)
Get verify flag	

Gets the current status of the verify flag. This flag determines whether or not data transmitted to a medium (floppy disk or hard drive) should be verified after the transmission.

Input	AH = 54H
Output	AL = Verify flag
	AL=0: Verify off
	AL=1: Verify on

Remarks

Function 2EH (see above) controls the status of the verify flag.

The contents of the AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES and flag registers are not affected by this function.

Interrupt 21H, Function 55H	DOS (Version 2.0 and above)
Create new PSP	

This function creates a new PSP by copying the indicated old PSP into the new one and then adding various new information into the new PSP. This new information is mainly the information which is dependent on the position of the PSP in storage and therefore can not be accepted unchanged from the old PSP into the new one.

Input	AH = 55H
	DX = Segment address of the new PSP
	CX = Segment address of the old PSP
Output	Carry-Flag=0: o.k.
	Carry-Flag=1: Error

Remarks

This undocumented function was created to be able to create a PSP for a program before loading the program. In view of the function 4BH, it is not required any more, because function 4BH loads the program and also automatically creates a new PSP that includes the adjustment of the Segment address in the PSP and in the program that was loaded.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 56H**DOS (Version 2.0 and above)**

Rename file (handle)

Renames a file or moves the file to another directory of a block device. Moving is possible only within the different directories of one particular device (i.e., you can't move a file from a hard drive directory to a floppy disk directory).

Input

AH = 56H

DS = Old filename segment address

DX = Old filename offset address

ES = New filename segment address

DI = New filename offset address

Output

Carry flag=0: O.K.

Carry flag=1: Error (AX = Error code)

AX=2: File not found

AX=3: Path not found

AX=5: Access denied

AX=11: Not the same device

Remarks

The directory name passed is an ASCII string which is terminated by an end character (ASCII code 0). It can contain a path designation and drive specifier. No wildcards are allowed. If no drive specifier or path designation exists, the function accesses the current drive or directory.

An error occurs if you attempt to move the file to a filled root directory.

This function cannot access subdirectories or volume names.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 57H, Subfunction 0**DOS (Version 2.0 and above)**

Get file date and time

Gets the date and time of the creation or last modification of a file.

Input

AH = 57H

AL = 0

BX = Handle

Output Carry flag=0: O.K.

CX=Time

DX=Date

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function

AX=6: Invalid handle

Remarks

For it to be accessed with a handle, the file must have been previously opened or created using one of the handle functions.

The time appears in the CX register in the following format:

Bits 0-4: Seconds in 2-second increments

Bits 5-10: Minutes

Bits 11-15: Hours

The date appears in the DX register in the following format:

Bits 0-4: Day of the month

Bits 5-8: Month

Bit 9-15: Year (relative to 1980)

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 57H, Subfunction 1

DOS (Version 2.0 and above)

Set file date and time

Sets the date and time of the creation or last modification of a file in the corresponding file and device.

Input AH = 57H

AL = 1

BX = Handle

CX = Time

DX = Date

Output Carry flag=0: O.K.

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function

AX=6: Invalid handle

Remarks

To be accessed with a handle, the file must have been previously opened or created using one of the handle functions.

The time appears in the CX register in the following format:

Bits 0-4: Seconds in 2-second increments

Bits 5-10: Minutes

Bits 11-15: Hours

The date appears in the DX register in the following format:

Bits 0-4: Day of the month

Bits 5-8: Month

Bit 9-15: Year (relative to 1980)

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 58H, Subfunction 0**DOS (Version 3.0 and above)**

Get allocation strategy

Determines the method currently in use by MS-DOS for allocating blocks of memory. If a program allocates memory using function 48H, different programs in memory may already have memory blocks assigned to them. Since these requested memory blocks vary in size, DOS has three methods of allocating memory to a program:

First fit: DOS starts searching at the start of memory and allocates the first memory block it finds of the requested size.

Best fit: DOS searches all available memory blocks and allocates the smallest suitable memory block it finds (the most efficient method).

Last fit: DOS starts searching at the end of memory and allocates the first memory block it finds of the requested size.

Input

AH = 58H

AL = 0

Output

Carry flag=0: O.K.

AX=0: First fit (start from beginning of memory)

AX=1: Best fit (search for best-fitting memory block)

AX=2: Last fit (start from end of memory)

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function number

Remarks

The allocation strategy applies to all programs.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 58H, Subfunction 1	DOS (Version 3.0 and above)
Get allocation strategy	

Defines the method used by MS-DOS for allocating blocks of memory. If a program allocates memory using function 48H, different programs in memory may already have memory blocks assigned to them. Since these requested memory blocks vary in size, DOS has three methods of allocating memory to a program:

First fit: DOS starts searching at the start of memory and allocates the first memory block it finds of the requested size.

Best fit: DOS searches all available memory blocks and allocates the smallest suitable memory block it finds (the most efficient method).

Last fit: DOS starts searching at the end of memory and allocates the first memory block it finds of the requested size.

Input

AH = 58H

AL = 1

BX = Allocation strategy

BX=0: First fit (start from beginning of memory)

BX=1: Best fit (search for best-fitting memory block)

BX=2: Last fit (start from end of memory)

Output

Carry flag=0: O.K.

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function number

Remarks

The allocation strategy applies to all programs.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 58H, Subfunction 02H	DOS (Version 5.0 and above)
Query the inclusion of the UMBs	

Starting with the version 5.0, the Upper-Memory-Blocks (UMBs), which are located between 640 KB and the 1-MB border, can be included in the DOS storage administration. This function informs the caller, if the UMBs are participating at this time in the storage administration.

Input

AH = 58H

AL = 02H

Output

Carry-Flag=0: o.k., in this case

AL = 0 : UMBs are not used

AL = 1 : UMBs are included in the storage administration

Carry-Flag=1: Error, in this case

AX = 1: no UMB support, because the MS-DOS=UMB command was not indicated

AX = 7: storage administration destroyed

Remarks

The error-code 7 is returned when DOS notices an inconsistency in its storage administration, which is usually caused by an erroneous access in a DOS program to this storage area.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 58H, Subfunction 03H	DOS (Version 5.0 and above)
Determine inclusion of the UMBs	

Starting with the Version 5.0, the Upper-Memory-Blocks (UMBs), which are located between 640 KB and the 1-MB border, can be included in the DOS storage administration. This function determines if the UMBs are participating at this time in the storage administration.

Input

AH = 58H

AL = 03H

BX = 0 : UMBs not participating

1 : UMBs are participating in the storage administration

Output

Carry-Flag=0: o.k.

Carry-Flag=1: Error, in this case

AX = 1: no UMB support, because the MS-DOS=UMB command was not indicated

AX = 7: storage administration destroyed

Remarks

The error-code 7 is returned when DOS notices an inconsistency in its storage administration, which is usually caused by an erroneous access in a DOS program to this storage area.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 59H	DOS (Version 3.0 and above)
Get extended error information	

Gets information about errors that occur during the call of one of the functions of either interrupt 21H or interrupt 24H. This information includes detailed information about the error, its origin and the action the user should take to alleviate the error.

Input

AH = 59H

BX = 0

Output

AX = Description of error

BH = Cause of error

BL = Recommended action

CH = Source of error

Remarks

The following codes describe the error:

Code	Error	Code	Error
0:	No error	20:	Unknown device
1:	Invalid function number	21:	Device not ready
2:	File not found	22:	Unknown command
3:	Path not found	23:	CRC error
4:	Too many files open at once	24:	Bad request structure length
5:	Access denied	25:	Seek error
6:	Invalid handle	26:	Unknown medium type
7:	Memory control block destroyed	27:	Sector not found
8:	Insufficient memory	28:	Printer out of paper
9:	Invalid memory address	29:	Write error
10:	Invalid environment	30:	Read error
11:	Invalid format	31:	General failure
12:	Invalid access code	32:	Sharing violation
13:	Invalid data	33:	Lock violation
14:	Reserved	34:	Unauthorized disk change
15:	Invalid drive	35:	FCB not available
16:	Current directory cannot be removed	80:	File already exists
17:	Different device	81:	Reserved
18:	No additional files	82:	Directory cannot be created
19:	Medium write protected	83:	Terminate after call of interrupt 24H

The following codes describe the cause of the error:

Code	Error	Code	Error
1:	No memory available on the medium	7:	Application program error
2:	Temporary access problem-may end soon	8:	File not found
3:	Access unauthorized	9:	Invalid file format/type
4:	Internal error in system software	10:	File locked
5:	Hardware error	11:	Wrong medium in drive, bad disk or medium problem
6:	Software failure not caused by running application program	12:	Other error

The following codes describe the source of the error:

Code	Error	Code	Error
1:	Unknown	4:	Serial device
2:	Block device (disk drive, hard drive, etc.)	5:	RAM
3:	Network		

The contents of the CS, DS, SS and ES registers are not affected by this function. All other register contents are destroyed.

Interrupt 21H, Function 5AH

DOS (Version 3.0 and above)

Create temporary file (handle)

Creates a temporary file in memory for storage during program execution. The filename doesn't matter because the access occurs through the assigned handle. Since this function allows several files open at the same time, DOS creates filenames from the current date and time. Every temporary file is ensured its own particular name because the function cannot be called more than once at a time.

Input

AH = 5AH

CX = File attribute

DS = Directory segment address

DX = Directory offset address

Output

Carry flag=0: O.K.

AX=Handle

DS=Complete filename segment address

DX=Complete filename offset address

Carry flag=1: Error (AX = Error code)

AX=3: Path not found

AX=5: Access denied

Remarks

The directory name passed is an ASCII string which is terminated by an end character (ASCII code 0). It can contain a path designation and drive specifier. No wildcards are allowed. If no drive specifier or path designation exists, the function accesses the current drive or directory.

The bits of the file attribute have the following meanings:

Bit 0 = 1: Read only file

Bit 1 = 1: Hidden file

Bit 2 = 1: System file

Temporary files are not automatically deleted after program execution. The file must be closed using function 3EH, then the temporary file must be deleted using function 41H.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function.

Interrupt 21H, Function 5BH**DOS (Version 3.0 and above)**

Create new file (handle)

Creates a file in the specified directory based upon an ASCII file format. If no drive specifier or path is provided, the file opens in the default (current) directory.

Input

AH = 5BH

CX = File attributes:

CX=00: Normal file

CX=01: Read-only file

CX=02: Hidden file

CX=04: System file

DS = ASCII file specification segment address

DX = ASCII file specification offset address

Output

Carry flag=0 (AX= file handle)

Carry flag=1 (AX = Error code)

AX=3: Path not found

AX=4: No handle available

AX=5: Access denied

AX=80 (50H): File already exists

Remarks

An error occurs when any element of the path designation doesn't exist, when the filename already exists in the specified directory, or when an attempt is made to create the file in an already full root directory.

The file defaults to the normal read/write attribute, which allows both read and write operations. This attribute can be changed by using function 43H.

Interrupt 21H, Function 5CH**DOS (Version 3.0 and above)**

Control record access

Locks or unlocks a particular section of a file. This function operates on multitasking and networking systems.

Input

AH = 5CH

AL = Function code

AL=00: Lock file section

AL=01: Unlock file section

BX = File handle
 CX = High word of section offset
 DX = Low word of section offset
 SI = High word of section length
 DI = Low word of section length

Output

Carry flag=0: Successful lock/unlock
 Carry flag=1: Error (AX = Error code)
 AX=1: Invalid function code
 AX=6: Invalid handle
 AX=33 (21H): All or part of section already locked

Remarks

This function can only be used on files already opened or created using functions 3CH, 3DH, 5AH or 5BH.

The corresponding call to unlock a file region must contain the identical file offset and file region length.

Interrupt 21H, Function 5CH, Subfunction 01H**DOS (Version 3.0 and above)**

Release of a blocked area in a file

With this function, areas in a file which had been blocked previously with the help of subfunction 00H, can be released again.

Input

AH = 5CH
 AL = 01H
 BX = Handle of the file
 CX = Hi-word of the address of the first byte in the file, which should be released again
 DX = Lo-Word, of the address of the first byte to be released
 SI = Hi-Word of the number of bytes to be released
 DI = Lo-Word of the number of bytes to be released
 Carry-Flag=0: o.k.
 Carry-Flag=1: Error, in this case AX = Error-Code
 1: Network software is not active
 6: invalid Handle
 33: the area indicated is not blocked

Remarks

This function can only be called after the SHARE-command or other network-software was previously called.

The Start-offset of the region to be blocked and its length are indicated as positive LONGINTS, which consist of 2 words and therefore 4 bytes. They must agree exactly with the indications that were made during a preceding call of the subfunction 00H.

The contents of the BX, CX, DX, SI, DI, BP and the segment registers are not affected by this function.

Interrupt 21H, Function 5DH	DOS (Version 1.0 and above)
Reserved	

Interrupt 21H, Function 5EH, Subfunction 0	DOS (Version 3.1 and above)
Get machine name	

Returns the address of an ASCII string which defines the local computer type within a network.

Input

AH = 5EH

AL = 00

DS = User buffer segment address

DX = User buffer offset address

Output

Carry flag=0: Successful execution

CH = 00: Name undefined

CH > 00: Name defined

CL = NETBIOS name number (when CH<>00)

DS = Identifier segment address (when CH<>00)

DX = Identifier offset address (when CH<>00)

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function code

Remarks

The computer type is a 15-byte-long string terminated by an end character (ASCII code 0).

Interrupt 21H, Function 5EH, Subfunction 2	DOS (Version 3.1 and above)
Set printer setup	

Specifies a string which precedes all output to a particular printer used by a network. This string allows network users to assign their own individual printing parameters to the shared printer.

Input

AH = 5EH

AL = 02

BX = Redirection list index (see Remarks below)

CX = Printer setup string length

DS = Printer setup string segment address

SI = Printer setup string offset address

Output Carry flag=0: Successful execution

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function code

Remarks

The contents of register BX (redirection list index) come from function 94 5EH, subfunction 2. Function 5EH, subfunction 3 (see below) can supply the current printer setup string.

Interrupt 21H, Function 5EH, Subfunction 3

DOS (Version 3.1 and above)

Get printer setup

Gets the printer setup string assigned to a particular network printer by using function 5EH, subfunction 2 (see above).

Input

AH = 5EH

AL = 03

BX = Redirection list index)

DS = Setup string receiving buffer segment address

SI = Setup string receiving buffer offset address

Output Carry flag=0: Successful execution

CX=Printer setup string length

ES=Segment address of buffer retaining setup string

DI=Offset address of buffer retaining setup string

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function code

Remarks

The contents of register BX (redirection list index) come from function 5EH, subfunction 2. Function 5EH, subfunction 3 can supply the current printer setup string.

Interrupt 21H, Function 5FH, Subfunction 2

DOS (Version 3.1 and above)

Get redirection list entry

Gets the system redirection list. This list assigns local names to network printers, files or directories.

Input

AH = 5FH

AL = 02

BX = Redirection list index (see Remarks below)
 DS = Device name buffer segment address (16 bytes)
 SI = Device name buffer offset address (16 bytes)
 ES = Network name buffer segment address (128 bytes)
 DI = Network name buffer offset address (128 bytes)

Output

Carry flag=0: Successful execution

BH = Status flag

BH=0: Valid device

BH=1: Invalid device

BL = Device type

BL=3: Printer

BL=4: Drive

BP = Destroyed

CX = Parameter value in memory

DX = Destroyed

DS = ASCII format local device name segment address

SI = ASCII format local device name offset address

ES = ASCII format network name segment address

DI = ASCII format network name offset address

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function code

AX=18: No more files available

Remarks

The contents of register CX come from function 5FH, subfunction 3 (see below).

Interrupt 21H, Function 5FH, Subfunction 3**DOS (Version 3.0 and above)**

Redirect device

Redirects device access in a network, assigning a network name to a local device.

Input

AH = 5FH

AL = 03

BL = Device type

BL=3: Printer

BL=4: Drive

CX = Parameter value in memory

DS = ASCII format local device name segment address

SI = ASCII format local device name offset address

ES = ASCII format network name and password segment address

DI = ASCII format network name and password offset address

Output

Carry flag=0: Successful execution

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function code; string format incorrect; device redirected

AX=3: Path not found

AX=5: Access denied

AX=8: Insufficient memory

Remarks

The contents of register CX are taken from function 5FH, subfunction 3.

Device names can be drive specifiers (e.g., A:), printer names (i.e., LPT1, PRN, LPT2 or LPT3) or null strings. If you enter a null string and password as the device name, DOS tries to open access to the network using the password.

Interrupt 21H, Function 5FH, Subfunction 4**DOS (Version 3.0 and above)**

Cancel redirection

Disables the current redirection by removing local name assignments to network printers, files or directories.

Input

AH = 5FH

AL = 04

BX = Redirection list index (see Remarks below)

DS = ASCII format local device name segment address

SI = ASCII format local device name offset address

Output

Carry flag=0: Successful execution

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function code; device name not on network

AX=15: Redirection halted

Remarks

Device names can be drive specifiers (e.g., A:), printer names (i.e., LPT1, PRN, LPT2 or LPT3) or strings beginning with double backslashes (i.e., \\). A string preceded by two backslashes terminates communications between the local computer and the network.

Interrupt 21H, Function 62H**DOS (Version 3.0 and above)**

Get PSP address

Gets the segment address of the PSP from the currently executing program.

Input

AH = 62H

BX = PSP segment address

Remarks

The PSP starts at address BX:0000.

The contents of the AX, CX, DX, SI, DI, BP, CS, DS, SS, ES registers and the flag registers are not affected by this function.

Interrupt 21H, Function 63H, Subfunction 0**DOS (Version 2.25 only)**

Get lead byte table

Gets the address of the system table which defines the byte ranges for the PC's extended character sets.

Input

AH = 9963H

AL = 00: Get address of system lead byte table

Output

DS = Table segment address

SI = Table offset address

Remarks

This function is available only in DOS Version 2.25.

Interrupt 21H, Function 63H, Subfunction 1**DOS (Version 2.25 only)**

Set or clear interim console flag

Clears the interim console flag.

Input

AH = 63H

AL = 01: Clear or set interim console flag

DL = Interim console flag setting

DL=01: Set interim console flag

DL=00: Clear interim console flag

Output

No output

Remarks

This function is available only in DOS Version 2.25.

Interrupt 21H, Function 63H, Subfunction 2	DOS (Version 2.25 only)
Gets the interim console flag	

Gets the interim console flag.

Input AH = 63H
 AL = 02: Get interim console flag value

Output DL = Flag value

Remarks

This function is available only in DOS Version 2.25.

Interrupt 21H, Function 64H	DOS (Version 3.0 and above)
Reserved	

Interrupt 21H, Function 65H	DOS (Version 3.3 and above)
Get extended country information	

Gets information about the specific country/code page.

Input AH = 65H
 AL = subfunction:
 AL = 1: Get international information
 AL = 2: Get uppercase pointer table
 AL = 4: Get pointer to uppercase pointer table (filename)
 AL = 6: Get pointer to collation table
 BX = Code page:
 BX = -1: active CON device
 CX = Length of buffer allocated to receive information
 DX = Country ID number
 DX = -1: Default
 ES:DI = Address of buffer allocated to receive information

Output Carry flag=0: Successful execution
 Carry flag=1: Error (AX = Error code)

Remarks

The information this function returns is an extended version of the information returned by int 21H, function 38H.

An error may occur if the country code in DX is invalid, or if the code page number is different from the country code, or if the buffer length specified in the CX register is less than five bytes. If the buffer is not long enough to receive all the information, the function accepts as much information as the buffer will accept. This buffer contains the following information after the call:

Byte 0: ID code for information

Bytes 1-2: Length of buffer

Bytes 3-4: Country ID

Bytes 5-6: Code page

Bytes 7-8: Date format

0 = USA: Month-day-year

1 = Europe: Day-month-year

2 = Japan: Year-month-day

Bytes 9-13: Currency indicator

Bytes 14-15: ASCII code of the thousand character (comma/period)

Bytes 16-17: ASCII code of the decimal character (period/comma)

Bytes 18-19: ASCII code of the date separation character

Bytes 20-21: ASCII code of the time separation character

Byte 22: Currency format

bit 0 = 0: Currency symbol before the value

bit 0 = 1: Currency symbol after the value

bit 1 = 0: No spaces between value and currency symbol

bit 1 = 1: Space between value and currency symbol

Byte 23: Precision (number of decimal places)

Byte 24: Time format

bit 0 = 0: 12-hour clock

bit 0 = 1: 24-hour clock

Bytes 25-28: Address of character conversion routine

Bytes 29-30: ASCII data separator

Bytes 31-40: Reserved

Interrupt 21H, Function 66H**DOS** (*Version 3.3 and above*)

Get or set code page

Gets or sets the current code page.

Input

AH = 66H

AL = subfunction:

AL = 1: Get code page

AL = 2: Select code page

BX = Selected code page (if AL = 2)

Output

Carry flag=0: Successful execution

If AL = 1 used for input:

BX = active code page

DX = default code page

Carry flag=1: Error (AX = Error code)

Remarks

If subfunction 2 is used, COUNTRY.SYS supplies the code page number.

The DEVICE... (CONFIG.SYS), NLSFUNC and MODE CP PREPARE commands (AUTOEXEC.BAT) must have already configured the system for code page switching before this function may be called.

Interrupt 21H, Function 67H**DOS** (*Version 3.3 and above*)

Set handle count

Sets the maximum number of accessible files and devices that may be currently opened using handles.

Input

AH = 67H

BX = Number of handles desired

Output

Carry flag=0: Successful execution

Carry flag=1: Error (AX = Error code)

Remarks

The PSP's default table reserved for the process can control 20 handles.

An error occurs if the content of the BX register is greater than 20, or if insufficient memory exists to allocate a block for the extended table.

If the number in the BX register is greater than the number of entries assigned by the FILES entry in the CONFIG.SYS file, no error occurs. However, attempts at opening a file or device fail if all file entries are in use, even if file handles are still available.

Interrupt 21H, Function 68H	DOS (Version 3.3 and above)
Commit file	

Writes all DOS buffers associated to a specific handle to the specified device. If the handle points to a file, the file's contents, date and size are updated.

Input AH = 68H
 BX = File handle

Output Carry flag=0: Successful execution
 Carry flag=1: Error (AX = Error code)

Remarks

This function performs the same task as closing and reopening a file or duplicate handle, even without handles. If this function accesses a character device's handle, the carry flag returns 0 but nothing else happens.

Multiprocessing and networking applications maintain control of the file.

Interrupt 21H, Function 69H	DOS (Version 1.0 and above)
Reserved	

Interrupt 21H, Function 6AH	DOS (Version 1.0 and above)
Reserved	

Interrupt 21H, Function 6BH	DOS (Version 1.0 and above)
Reserved	

Interrupt 21H, Function 6CH	DOS (Version 4.0 and above)
Extended Open function	

Starting with Version 4.0 of DOS there is the capability to create a file during the OPEN-call, or to overwrite an already existing file. The function therefore enhances the capability of the normal OPEN function 3DH.

Input AH = 3DH
 AL = 0
 BX = access mode
 CX = file attribute
 DX = DOS reaction
 DS:SI = FAR-pointer to the buffer with the filename

Output Carry-Flag=0: o.k., in this case AX = Handle of the file
 CX = Status

Carry-Flag=1: Error, in this case AX = Error-Code

- 1: File-Sharing software missing
- 2: file not found
- 3: Path not found or file does not exist
- 4: no more free Handles
- 5: access denied
- 12: this access mode not allowed

Remarks

The filename must be available as an ASCII string which is terminated with an end character (ASCII-Code 0). Besides a device designation it may contain a complete Path designation and a filename, but not a Wildcard. If the device designation or Path description is missing, access occurs to the current device or the current Directory.

The access mode is constructed as follows:

Bit 0 - 2: Read/Write permission

- 000b = file can only be read
- 001b = file can only be written
- 010b = file can be read and written

Bit 3: 0b

Bit 4 - 6: File-Sharing mode

- 000b = only the current program may access the file (Compatibility mode)
- 001b = only the current program may access the file
- 010b = another program may read, but not write the file
- 011b = another program may write, but not read the file
- 100b = another program may read and write the file

Bit 7: Handle-Flag

- 0 = Also the child-program of the current program is allowed to access the Handle of this file.
- 1 = Only the current program can access the Handle of this file.

Bit 8 - 12: 0b

Bit 13: 0 = On a critical error call Interrupt 24h.

- 1 = On a critical error return the corresponding error code to AX, but do not call Interrupt 24h.

Bit 14 1 = For every write access to this file immediately change the Directory entry.

Bit 15: 0b

The file attribute in the CX-Register can be loaded before the function call with the following attributes:

Bit 0 = 1: file can only be read, but not written (Read-Only)

Bit 1 = 1: file is hidden (not displayed during DIR command)

Bit 2 = 1: file is a system-file

Bit 5 = 1: file has been changed since the last archiving

Bit 6-15 : 0b

The reaction of the function is stored in the lower two nibbles of the DX-Register, if the file to be opened does not exist or does exist. Bits 0 to 3 represent the reaction if the file does not yet exist. The values mean:

0000b = Terminate function with an error-code

0001b = Create file

In bits 4 to 7 the reaction to an already existing file is recorded. The values have the following meaning:

0000b = Terminate function with an error-code

0001b = Open existing file

0010b = Overwrite existing file and open

The function 3DH can continued to be used to open a file also under DOS 4.0.

If the function call was terminated in an orderly manner, all other Handle functions can be called through the passed Handle. In the CX register remains the function status which provides information about the action of the function as follows:

0 = File opened

1 = New file created and opened

2 = Existing file overwritten and opened

The file pointer is set to the first byte of the file.

The File-Sharing mode in bits 4 to 6 of the access mode is also under DOS version 4 only interesting if the file is on a mass storage device which is part of a network. In this case, these 3 bits decide, if other programs in the network may access this file during the opening and the access rights of this program. If the value 0 is indicated for these bits, DOS treats this file in the Compatibility-Mode, in which the existence of a network in relation to this file is ignored and only the current program may access this file.

Error 12 can only occur under DOS version 3 and only in a network, if the file was already opened by another program and it was determined at that time that at this moment no other program may access it.

The contents of the BX, DX, SI, DI, BP and segment registers are not affected by this function.

Interrupt 22H

DOS (Version 1.0 and above)

Terminate address

Contains the address of a routine which terminates a program. Control returns to the program that called for termination. You should never call this routine directly.

DOS stores the contents of this interrupt vector in the PSP of the program to be executed before passing control to the program. This prevents program changes to the vector, which could prevent DOS from calling the termination routine.

Interrupt 23H**DOS (Version 1.0 and above)****Ctrl Break** handler address

Contains the address of a routine which executes when the user presses **Ctrl C** or **Ctrl Break**. You should never directly call this routine.

DOS stores the contents of this interrupt vector in the PSP of the program to be executed before passing control to the program. This prevents program changes to the vector, which could prevent DOS from calling the termination routine.

Interrupt 24H**DOS (Version 1.0 and above)**

Critical error handler address

Represents a routine called during hardware access (e.g., disk drive) when a critical error occurs. You should never directly call this routine.

When an application routine is called during a critical error, bit 7 of the AH register indicates the type of failure (0 = disk/hard drive error, 1 = other errors). A disk/hard disk error will only be reported after several attempted accesses. During the call, the DI register receives one of the following codes:

0:	Disk write protected	7:	Unknown device type
1:	Access on unknown device	8:	Sector not found
2:	Drive not ready	9:	Printer out of paper
3:	Invalid command	10:	Write error
4:	CRC error	11:	Read error
5:	Bad request structure length	12:	General failure
6:	Seek error		

The error routine restores the SS, SP, DS, ES, BX, CX and DX registers to the same values that they contained during the call. During execution it can only access functions 1 to 0CH of interrupt 21H. It should be terminated by an IRET instruction and pass one of the following codes to the AL register:

0:	Ignore error	2:	Terminate program using interrupt 23H
1:	Repeat the operation	3:	Fail system call (Version 3 and up only)

If a program changes the content of this interrupt vector, the program can terminate without restoring the memory contents. Since RAM can be released and used by other programs, the critical error routine can be overwritten by another program in memory. When this occurs, a critical error could cause a system crash because a completely different code now exists at the location of the old error handler routine.

Before passing control to the program, DOS stores the contents of this interrupt vector in the PSP of the program to be executed. This prevents program changes to the vector, which could prevent DOS from calling the termination routine. During program termination, the contents of the interrupt vector pass from the PSP to the vector; then the system calls the routine.

Interrupt 25H	DOS (Version 1.0 and above)
Absolute disk read	

Reads one or more consecutive sectors from a disk or hard drive.

Input

AL = Drive specifier
CX = Number of sectors to read
DX = First sector to read
DS = Buffer segment address
BX = Buffer offset address

Output

Carry flag=0: O.K.
Carry flag=1: Error (AX = Error code)
AX=1: Bad command
AX=2: Bad address
AX=4: Sector not found
AX=8: DMA error
AX=16: CRC error
AX=32: Disk controller error
AX=64: Seek error
AX=128: Device does not respond

Remarks

In the AL register 0 represents drive A:, 1 represents drive B:, etc.

All the sectors of the medium can be accessed. DOS itself uses this interrupt to read the root directory and the FAT of a medium. The data are read from the medium into the buffer of the calling program. After the function call, the contents of all registers, except the segment register, may change.

After the interrupt call, the stack pointer changes position because two bytes stored on the stack during the call are removed and not returned. These bytes represent the flag register, which can be read from the stack using the POPF instruction. The old value of the stack pointer can be set by adding 2 to its contents. If you omit the stack pointer correction, the stack could overflow. Because of this, you cannot call this interrupt from higher level languages. You must call it from assembly language.

The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function. The contents of all other registers may change.

Interrupt 26H	DOS (Version 1.0 and above)
Absolute disk write	

Writes one or more consecutive sectors to a disk or hard drive.

Input

AL = Device designation
CX = Number of sectors to be written
DX = First sector to be written
DS = Buffer segment address
BX = Buffer offset address

Output

Carry flag=0: O.K.
Carry flag=1: Error (AX = Error code)
AX=1: Bad command
AX=2: Bad address
AX=3: Medium write protected
AX=4: Sector not found
AX=8: DMA error
AX=16: CRC error
AX=32: Disk controller error
AX=64: Seek error
AX=128: Device does not respond

Remarks

In the drive specifier 0 represents drive A:, 1 represents drive B:, etc.

All the sectors of the medium can be accessed. DOS itself uses this interrupt to write the root directory and the FAT to a medium. The data are written from the buffer of the calling program to the medium. After the function call, the contents of all registers, except the segment register, may change.

After the interrupt call, the stack pointer changes position because two bytes stored on the stack during the call are removed and not returned. These bytes represent the flag register, which can be read from the stack using the POPF instruction. The old value of the stack pointer can be set by adding 2 to its contents. If you omit the stack pointer correction, the stack could overflow. Because of this, you cannot call this interrupt from higher level languages. You must call it from assembly language. The contents of the BX, CX, DX, SI, DI, BP, CS, DS, SS and ES registers are not affected by this function. The contents of all other registers may change.

Interrupt 27H**DOS (Version 1.0 and above)**

Terminate and stay resident

Terminates the currently executing program and returns control to the program that called the current program. Unlike other functions used for program termination, the memory used by the current program keeps the program code for later recall.

Input

CS = PSP segment address
DX = Number of bytes + 1 to be reserved

Output No output

Remarks

This function is only suitable for calling COM programs.

The number of bytes to be reserved relates to the beginning of the PSP.

The value in the DX register has no effect on memory blocks reserved by function 48H of interrupt 21H.

An error occurs during the call of this interrupt if the value in the DX register ranges from FFF1H to FFFFH.

This interrupt does not close open files.

Interrupt 2FH, Subfunction 0	DOS (Version 3.0 and above)
Get print spool intall status	

Gets current installation status of the print spooler.

Input AH = 2FH

AL = 0

Output Carry flag=0: Successful execution

AL = 0: O.K. to install

AL = 1: Don't install

AL = 255: Already installed

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function

AX=2: File not found

AX=3: Path not found

AX=4: Too many files currently open

AX=5: Access denied

AX=8: Print queue full

AX=9: Print spooler busy

AX=12: Name too long

AX=15: Invalid drive

Interrupt 2FH, Subfunction 1	DOS (Version 3.0 and above)
Send file to print spooler	

Passes a file to the print spooler.

Input

AH = 2FH
 AL = 1
 DS = Print packet (see below) segment address
 DX = Print packet (see below) offset address

Output

Carry flag=0: Successful execution
 Carry flag=1: Error (AX = Error code)
 AX=1: Invalid function
 AX=2: File not found
 AX=3: Path not found
 AX=4: Too many files currently open
 AX=5: Access denied
 AX=8: Print queue full
 AX=9: Print spooler busy
 AX=12: Name too long
 AX=15: Invalid drive

Remarks

The five-byte print packet contains print spooler information. The first byte indicates the DOS version (0=Versions 3.1 to 3.3); the remaining bytes indicate the segment and offset addresses of the file specification.

Interrupt 2FH, Subfunction 2**DOS (Version 3.0 and above)**

Remove file from print queue

Deletes a file from the print spooler queue.

Input

AH = 2FH
 AL = 2
 DS = ASCII-format file segment address
 DX = ASCII-format file offset address

Output

Carry flag=0: Successful execution
 Carry flag=1: Error (AX = Error code)
 AX=1: Invalid function
 AX=2: File not found
 AX=3: Path not found

AX=4: Too many files currently open

AX=5: Access denied

AX=8: Print queue full

AX=9: Print spooler busy

AX=12: Name too long

AX=15: Invalid drive

Remarks

This subfunction allows wildcards (? and *) in file specifications, allowing you to delete more than one file at a time from the print queue.

Interrupt 2FH, Subfunction 3

DOS (Version 3.0 and above)

Cancel all files in print queue

Cancels all files waiting in the print spooler queue for printing.

Input

AH = 2FH

AL = 3

Output

Carry flag=0: Successful execution

Carry flag=1: Error (AX = Error code)

AX=1: Invalid function

AX=2: File not found

AX=3: Path not found

AX=4: Too many files currently open

AX=5: Access denied

AX=8: Print queue full

AX=9: Print spooler busy

AX=12: Name too long

AX=15: Invalid drive

Interrupt 2FH, Subfunction 4

DOS (Version 3.0 and above)

Hold print jobs for status check

Halts all print jobs while testing for spooler status.

Input

AH = 2FH

AL = 4

Output

Carry flag=0: Successful execution

Carry flag=1: Error

DX = Number of errors

DS = Print queue segment address

SI = Print queue offset address

Remarks

The print queue segment and offset addresses point to a set of 64-byte filenames in the queue. Each entry contains an ASCII file specification.

The first filename in the queue is the file currently printing in the print spooler. The last filename in the queue has a zero in the first byte of the specification.

Multiplexer-Interrupt 2FH

Because interrupt 2FH represents an interface to various resident DOS programs and allows TSR programs to make their functions available to other programs, it is known as the multiplexer.

Interrupt 2FH, Code 01H, Function 00H	MUX
PRINT: Determine installation status	

Programs can use this function to determine whether the memory resident portion of the PRINT DOS command has been loaded.

Input AH = 01H

 AL = 00H

Output AL = FFH: PRINT installed

Remarks

If PRINT has been installed, this function will return the value FFH in the AL register. Only then can the other PRINT functions be called. This function call must therefore precede all other PRINT functions.

Interrupt 2FH, Code 01H, Function 01H	MUX
PRINT: Add file to wait list	

This function allows a program to add a file to the end of the printer wait list, to be printed from the background via PRINT.

Input AH = 00H

 AL = 01H

 DS = Segment address of FAR pointer to file info data structure

 DX = Offset address of FAR pointer to file info data structure

Output Carry flag = 0 : o.k.

 Carry flag = 1: Error, in this case

 AX = error code (see remarks)

Remarks

The data structure accessed through the pointer in DS:DX comprises 5 bytes and must adhere to the following structure:

Ofs	Meaning	Type
00H	Always 0	1 Byte
01H	FAR pointer to ASCIIZ string with filename	1 FAR pointer

Wildcards are not permitted in the specified filename. However, drive and path may be designated.

If an error occurs, the one of the following error codes will be returned:

0001H	Unknown function
0002H	File not found
0003H	Path not found
0004H	Too many files opened
0005H	File access denied
0008H	Printer wait list is full
000FH	Unknown device

This function should only be called after function 00H has indicated that the resident portion of PRINT has been installed.

Interrupt 2FH, Code 01H, Function 02H
MUX

PRINT: Delete from wait list

This function can be used to remove one or more files from the printer wait list. If the file currently being printed is also specified, print output will be stopped.

Input

AH = 01H

AL = 02H

DS = Segment address of pointer to filename as ASCIIZ string

DX = Offset address of pointer to filename as ASCIIZ string

Output

Carry flag = 0 : o.k.

Carry flag = 1: Error, in this case

AX = error code (see remarks)

Remarks

Wildcards may be used in the specified filename so that several files can be removed from the wait list at one time. The only possible error code here is 0002H, "file not found", in the event that the specified file was not included in the printer wait list. This function should only be called when function 00H has indicated that the resident portion of PRINT has been installed.

Interrupt 2FH, Code 01H, Function 03H
MUX

PRINT: Delete wait list

This function deletes all files from the printer wait list, and any current print job is stopped immediately.

Input AH = 01H

AL = 03H

Output none

This function should only be called when function 00H has indicated that the resident portion of PRINT has been installed.

Interrupt 2FH, Code 01H, Function 04H	MUX
PRINT: Stop printer output and get status	

This function can be used to determine the current contents of the printer wait list and the number of previous output errors.

Input AH = 01H

AL = 04H

Output DX = Error counter

DS:SI = FAR pointer to wait list

Remarks

Upon this function call the printer output is interrupted, and resumes after function 05H is called.

The pointer returned in DS:SI points to the buffer in which DOS retains the waiting list in a series of 64 byte entries. Each of these entries contains an ASCII string with the name of the file which is to be printed. To identify the list's end, its last entry always begins with the NUL character (00).

The contents of this list may be read, but not modified.

This function should only be called when function 00H has indicated that the resident portion of PRINT has been installed.

Interrupt 2FH, Code 01H, Function 05H	MUX
PRINT: Continue print job	

After a print job has been interrupted by function 04H, this function can be used to resume the print job.

Input AH = 01H

AL = 05H

Output none

Remarks

This function call is only useful if function 04H has previously been executed.

Interrupt 2FH, Code 01H, Function 06H	MUX
PRINT: Locate printer	

If the printer wait list contains one file, this function can be used to determine the printer to which the print job will be sent.

Input AH = 01H

AL = 06H

Output Carry flag = 0 : Wait list is empty
 Carry flag = 1: Wait list is not empty, in this case
 AX = 0008H
 DS:SI = FAR pointer to the head of the printer driver

Remarks

The values returned by this function are confusing, since the information that's needed (the pointer to the head of the printer device driver) is returned only with a set carry flag and the error code 0008 (wait list full). If the wait list is not empty, the carry flag will be cleared. In this case the pointer to the head of the printer driver will be unavailable.

Interrupt 2FH, Code 06H, Function 00H	MUX
PRINT: Get installation status	

This function allows a program to determine whether the resident portion of the DOS command ASSIGN has been loaded.

Input AH = 06H

AL = 00H

Output AL = FFH: ASSIGN installed

Interrupt 2FH, Code 10H, Function 00H	MUX
SHARE: Get installation status	

This function indicates whether the resident portion of the DOS command SHARE has been loaded.

Input AH = 10H

AL = 00H

Output AL = FFH: SHARE installed

Interrupt 2FH, Code 1AH, Function 00H	MUX
ANSI.SYS: Determine installation status	

This function indicates whether the device driver ANSI.SYS has been installed.

Input AH = 1AH

AL = 00H

Output AL = FFH: ANSI.SYS installed

Interrupt 2FH, Code 43H, Function 00H	MUX
HIMEM.SYS: Get installation status	

This function indicates whether the device driver HIMEM.SYS has been installed. This driver allows extended memory to be used in accordance with the XMS standard.

Input	AH = 43H
	AL = 00H
Output	AL = 80H: HIMEM.SYS installed

Remarks

Please note that this function, in contrast to most other functions that perform installation checks, returns the value 80H when successful, instead of the value FFH.

To maintain compatibility with HIMEM.SYS, this function call is also supported by other XMS memory managers.

Interrupt 2FH, Code 43H, Function 10H	MUX
HIMEM.SYS: Determine address for XMS function calls	

Unlike other memory managers, HIMEM.SYS calls XMS functions through a FAR CALL procedure, instead of an interrupt. This function is used to determine the address of that procedure.

Input	AH = 43H
	AL = 10H
Output	ES:BX = FAR pointer for calling XMS functions

Remarks

This function may only be called when a preceding call of function 00H has indicated that HIMEM.SYS has been installed.

Interrupt 2FH, Code 48H, Function 00H	MUX
DOSKEY: Get installation status	

This function indicates whether the DOS program DOSKEY.COM has been loaded.

Input	AH = 48H
	AL = 00H
Output	AL = 00H: not installed

Remarks

DOSKEY was introduced with DOS Version 5.0.

Please note that unlike other installation check functions, the only return value defined for this function is the value indicating that DOSKEY.COM has not been installed.

Interrupt 2FH, Code 48H, Function 10H	MUX
DOSKEY: Receive user input	

A program can access the DOSKEY program to prompt and receive user input.

Input

AH = 48H

AL = 10H

DS:DX = FAR pointer to data structure (see remarks)

Output none

Remarks

The data structure referenced through the pointer in DS:DX encompasses 130 bytes and receives the user input. It must be structured according to the following pattern:

Ofs	Meaning	Type
00h	Input buffer size, must be 128	1 Byte
01h	The number of character read minus 1	1 Byte
02h	The input buffer	128 Bytes

The number of bytes that have been read is entered into the data structure by DOSKEY. Therefore, unlike the length specification in the first byte, this field doesn't need to be initialized before DOSKEY is called.

Interrupt 2FH, Code 4A11H, Function 00H
DBLSPACE: Scan drive map

Enables a program to determine whether DoubleSpace is loaded and to obtain information about the DoubleSpace driver.

Input

AX = 4A11H

BX = 00H

Output

AX = 00H: DoubleSpace installed

BX = 444DH

CL = First drive designation used by DoubleSpace (65 = A:)

CH = Number of drive designations reserved for DoubleSpace

DX = Internal DoubleSpace version number

Remarks

If the function does not return 0 in register AX then DoubleSpace is not installed and the contents of the other return registers are undefined.

The value in BX stands for the two ASCII characters "MD", i.e., Microsoft DoubleSpace.

The two return values in CH and CL come from the FirstDrive and LastDrive settings in the DoubleSpace configuration file DBLSPACE.INI. Please note that, contrary to normal conventions, CL receives the value 65 (the ASCII code for A) for drive A:, rather than 0.

The internal version number from register DX is utilized by DBLSPACE.BIN, IO.SYS, and DBLSPACE.EXE, in order to maintain consistency. If bit 15 of this number is set, it is due to omitting the /MOVE parameter when calling the DBLSPACE.SYS driver in the CONFIG.SYS file. In this case, DoubleSpace stays in memory under 640K instead of shifting into upper memory.

Interrupt 2FH, Code 4A11H, Function 01H

DBLSPACE: Scan drive map

Determines whether a drive actually has another drive hidden behind it and whether a drive is compressed.

Input

AX = 4A11H
BX = 01H
DL = Drive to be scanned (A: = 0)

Output

AX = 00H: DoubleSpace installed
BL = Bit 7 = 1 : Compressed drive
= 0 : Uncompressed drive
Bits 0 to 6 : Number of host drive
BH = Number of CVF file (DBLSPACE.xxx), if drive is compressed

Remarks

If the function does not return 0 in register AX then DoubleSpace is not installed, and the contents of the other return registers are undefined.

If Bit 7 in register BL is set following the function call, then the drive is compressed. In that case a new call to this function must be made, with the value returned from register BL (bits 0 to 6) as the device code in DL, to ascertain the host drive. If the value from DL is again returned in bits 0 to 6 of BL, then the drive is a "swapped" drive. The host drive in this case is the drive from the first call (the return value in BL). In the other case, the drive designation has not been swapped and is therefore genuine. Here the second function call has returned the number of the host drive in bits 0 to 6 of register BL.

A drive designation for an uncompressed drive is genuine if bit 7 of BL is not set following the function call, and bits 0 to 6 of BL contain the same device number as was given in DL when the function was called.

Interrupt 2FH, Code 4A11H, Function 02H

DBLSPACE: Swap drive designations

This function arranges the swapping of a DoubleSpace device designation with that of its host drive.

Input

AX = 4A11H
BX = 02H
DL = Number of compressed drive whose drive designation is to be swapped (0 = A)

Output AX = 00H: Drive designations were successfully swapped

 101H: Invalid drive designation

 102H: Drive entered is not compressed

 103H: Drive designations already swapped

 All other values: DoubleSpace not installed

Remarks

This function is intended for internal DoubleSpace use only.

Interrupt 2FH, Code 4A11H, Function 03H

DBLSPACE: For internal use only

Interrupt 2FH, Code 4A11H, Function 04H

DBLSPACE: For internal use only

Interrupt 2FH, Code 4A11H, Function 05H

DBLSPACE: Mount compressed drive

Enables the mounting of a compressed drive into the system. The call is very complicated however, which is why Microsoft recommends a direct call to DBLSPACE.EXE using the DOS Exec function. The syntax here is as follows:

DBLSPACE.EXE /Mount=[CVFNumber] HostDrive: [/NEWDRIVE=NewDriveDesignation:]

Interrupt 2FH, Code 4A11H, Function 06H

DBLSPACE: Unmount a DoubleSpace drive

Unmounts an activated DoubleSpace drive, rendering it nonexistent from the user's perspective.

Input AX = 4A11H

 BX = 06H

 DL = Drive to be deactivated (0 = A:)

Output AX = 00H: Drive deactivated

 102H: Drive entered is not compressed and cannot be deactivated

Remarks

If the function returns none of these values in Register AX, then DoubleSpace isn't installed.

The compressed drive remains unchanged on the hard drive, yet DOS no longer recognizes its former device ID.

Interrupt 2FH, Code 4A11H, Function 07H

DBLSPACE: Establish storage space

This function returns the total number of sectors in the sector heap of a compressed drive, as well as the number still free.

Input

AX = 4A11H

BX = 07H

DL = Drive designation (0 = A:)

Output

AX = 00H: All OK

DS:SI = Refers to a Dword with the total number of sectors in the sector heap

DS:SI+4 = Refers to a Dword with the number of free sectors in the sector heap

Remarks

If the function does not return 0 in register AX then DoubleSpace is not installed and the contents of the other return registers are undefined.

Interrupt 2FH, Code 4A11H, Function 08H

DBLSPACE: Obtain information about CVF file fragmentation

Returns information regarding fragmentation of all CVF files on a particular host drive.

Input

AX = 4A11H

BX = 08H

DL = Compressed drive (0 = A:)

Output

AX = 00H: OK, in this case

BX = Maximum value for fragmentation

CX = Number of additional fragmentations permitted

AX = 102H: Drive entered is not compressed and cannot be deactivated

Remarks

If the function returns none of these values in Register AX then DoubleSpace is not installed and the contents of the other return registers are undefined.

Interrupt 2FH, Code 4A11H, Function 09H

DBLSPACE: Scan for maximum number of compressed drives

This function determines the number of DISK_UNIT structures which DOS needs for maintaining DoubleSpace drives, and which DoubleSpace creates in memory during the boot process for later definition of DoubleSpace drives.

Input

AX = 4A11H

BX = 09H

DL = Device ID of any compressed drive (0 = A:)

Output

AX = 0000H: OK, in this case

CL = Number of DISK_UNIT structures allocated by DoubleSpace during bootup.

Remarks

If the function does not return 0 in Register AX then DoubleSpace is not installed and the contents of the other return registers are undefined.

Each DISK_UNIT structure takes up 96 bytes of memory. The number returned here is determined by the MaxRemovableDrives parameter from the DoubleSpace initialization file DBLSPACE.INI.

Interrupt 2FH, Code ADH, Function 80H MUX

KEYB.COM: Return version number

This function returns the version number of the current DOS keyboard driver KEYB.COM.

Input AH = ADH

AL = 80H

BX = 0

Output BH = Main version number

BL = Sub version number

Remarks

If BH and BL are 00H after the function has been called, KEYB.COM is not installed.

Interrupt 2FH, Code ADH, Function 80H

MUX

KEYB.COM: Return version number

This function is used to set the active code page.

Input AH = ADH

AL = 81H

BX = Code page number

Output Carry flag = 0 : o.k.

Carry flag = 1: Error, in this case

AX = 0001 "unknown code page"

Output none

Remarks

The following code pages are recognized by KEYB:

Code	Character set	Code	Character set
437	USA	863	French Canadian
850	Multi-lingual (all European countries)	865	Scandinavian
860	Portuguese		

Before calling this function, you should use function 80H to determine whether KEYB.COM is active.

You'll find more information on code pages in your DOS manual. In the framework of PC System Programming these do not come into play.

Interrupt 2FH, Code B0H, Function 00H	MUX
GRAFTABL: Get installation status	

This function indicates whether the resident portion of the DOS command GRAFTABL has been loaded.

Input AH = B0H
 AL = 00H

Output AL = FFH: GRAFTABL loaded

Interrupt 2FH, Code B7H, Function 00H	MUX
APPEND: Get installation status	

This function determines whether the resident portion of the DOS command APPEND has been loaded.

Input AH = B7H
 AL = 00H

Output AL = FFH: APPEND loaded

Interrupt 2FH, Code B7H, Function 02H	MUX
APPEND: Verify DOS 5 compatibility	

Input AH = B7H
 AL = 02H

Output AX = FFFFH : DOS 5.0 compatible

Remarks

This function may only be called when function 00H has indicated that APPEND has been loaded.

Interrupt 2FH, Code B7H, Function 04H	MUX
APPEND: Get list of APPEND directories	

This function returns the various directories that have been specified as the search path for files using APPEND.

Input AH = B7H

AL = 04H

Output ES:DI = FAR pointer to the buffer containing the APPEND directories.

Remarks

The buffer identified by the pointer in ES:DI, returned by the function, contains the various APPEND directories as ASCIIZ strings. As with the PATH command, the individual directories are separated by semicolons. The contents of this buffer must not be changed by the user.

This function may be called only when function 00H has indicated that APPEND has been loaded.

Interrupt 2FH, Code B7H, Function 06H

MUX

APPEND: Determine operation mode

This function returns the operation mode specified in the DOS command line at the APPEND call, using the different function switches.

Input AH = B7H

AL = 06H

Output BX = APPEND flag

Remarks

The individual bits of the returned flag represent the different operational modes. Bits that are not included in the following chart has no meaning and contain the value 0.

Bit	Description
0	1 = APPEND is active
12	1 = the APPEND directories will be included in the search only if a drive is specified with the file that is to be found, and if that drive specification corresponds to the drive declared in that particular APPEND string.
13	1 = the switch /PATH:ON is active
14	1 = the switch /E is active
15	1 = the switch /X:ON is active

This function may be called only when function 00H has indicated that APPEND has been loaded.

Interrupt 2FH, Code B7H, Function 07H

MUX

APPEND: Set operation mode

This function is the counterpart of function 06H and sets the different APPEND options.

Input

AH = B7H

AL = 07H

BX = APPEND flag

Output

none

Remarks

The bit values and their meaning for the append flag can be taken from the chart listed for function 06H.

This function may be called only when function 00H has indicated that APPEND has been loaded.

EMM Functions

The EMS standard which was introduced by Lotus, Intel and Microsoft. It defines the access to the memory expansion cards which operate on the "bank switching" principle which only accesses a small part of the whole memory expansion.

Most EMS cards in circulation support version 3.2 of this standard and therefore the EMS-functions, which were defined under Version 3.0 and 3.2. Only a few memory cards support the EMS-specification 4.0 and the functions connected with it. This is not true for most of the commercial EMS emulators, which simulate the EMS memory with Extended Memory. They're usually similar to version 4.0.

Interrupt 67H, Function 40H	LIM/EMS
Expanded memory: Get status	

Returns the error status of the EMM after calling any EMS functions.

Input	AH = 40H
Output	AH = EMM status
	AH=00H: O.K.
	AH=80H: Internal error, EMM possibly destroyed
	AH=81H: EMS hardware error

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information). This function should be the first EMM call a program makes, to ensure that the hardware and software are functioning properly.

Interrupt 67H, Function 41H	LIM/EMS
Expanded memory: Get segment address of the page frame	

Determines the segment address of the page frame.

Input	AH = 41H
Output	AH = 0: O.K.
	BX = Page frame segment address
	AH > 0: Error
	AH=80H: Internal error, EMM possibly destroyed
	AH=81H: EMS hardware error

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information). The addresses of the four physical pages can be calculated from this segment address, whereby the first page starts at address PAGE_FRAME:0000. The three other pages follow at 16K intervals.

Interrupt 67H, Function 42H	LIM/EMS
Expanded memory: Get number of EMS pages	

Informs the calling program how many 16K EMS pages are installed, and how many EMS pages are still available or unallocated.

Input AH = 42H

Output AH = 0: O.K.
 BX = Number of free (unallocated) pages
 DX = Total number of EMS pages
 AH > 0: Error
 AH=80H: Internal error, EMM possibly destroyed
 AH=81H: EMS hardware error

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information). The number of kilobytes of free EMS memory can be calculated by multiplying the number of free pages by 16.

Interrupt 67H, Function 43H	LIM/EMS
Expanded memory: Allocate EMS memory	

Allocates a given number of 16K EMS pages for later access.

Input AH = 43H
 BX = Number of logical (16K) pages to be allocated

Output AH = 0: O.K.
 DX = Handle for accessing allocated memory
 AH > 9: Error
 AH=80H: Internal error, EMM possibly destroyed
 AH=81H: EMS hardware error
 AH=85H: No more handles available
 AH=87H: Not enough pages free
 AH=88H: No pages were requested

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information).

The handle returned can be used for future access and for releasing the allocated memory. If this handle is "lost", the handle cannot be recovered, nor can memory be released or used by other programs.

A call to this function may fail because there are not enough pages free or because the EMM has been called so often that no more handles are available.

The handles normally have the numbers FF00H, FE01H, FD02H, FC03H, etc.

Interrupt 67H, Function 44H	LIM/EMS
Expanded memory: Set mapping	

Places one of the pages previously allocated by function 43H in one of the four physical pages within the page frame.

Input

AH = 44H

AL = Physical page number (0 to 3)

BX = Logical page number

DX = Handle

Output

AH = Error status

AH=00H: O.K.

AH=80H: Internal error, EMM possibly destroyed

AH=81H: EMS hardware error

AH=83H: Invalid handle

AH=8AH: Invalid logical page

AH=8BH: Invalid physical page

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information).

The handle used when calling this function must have been returned by a previous call to EMM function 43H.

The logical pages are numbered from 0 on, so that the value 0 must be passed to access the first logical page. The largest value allowed is the number of allocated pages minus one.

Before accessing the physical page, the segment address of the page frame must be determined with function 41H.

Interrupt 67H, Function 45H	LIM/EMS
Expanded memory: Release pages	

Releases pages allocated with function 43H to the EMM. This makes these pages available to other applications.

Input AH = 45H

DX = Handle

Output AH = Error status:

AH=00H: O.K.

AH=80H: Internal error, EMM possibly destroyed

AH=81H: EMS hardware error

AH=83H: Invalid handle

AH=85H: Error while saving and restoring mapping

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information).

The handle used when calling this function must have been returned by a previous call to EMM function 43H.

All the pages allocated to this handle are released by this function. It is impossible to release individual pages.

After a successful call to this function the handle is no longer valid and cannot be used for accessing EMS memory.

If the function returns an error, you should repeat the call at least three times or the pages will remain allocated and will not be available for other programs.

Interrupt 67H, Function 46H	LIM/EMS
Expanded memory: Get EMM version	

Determines the version number of the EMM (Expanded Memory Manager).

Input AH = 46H

Output AH = 0: O.K.

AL = EMM version number

AH > 0: Error

AH=80H: Internal error, EMM possibly destroyed

AH=81H: EMS hardware error

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information).

The EMM version number is stored in the AL register as a BCD number, in which the upper four bits represent the version number preceding the decimal point and the lower four bits represent the version number following the decimal point. See also the demonstration programs in Chapter 12.

Interrupt 67H, Function 47H	LIM/EMS
Expanded memory: Save mapping	

Saves current mapping between the four physical pages in the page frame and the associated logical pages.

Input	AH = 47H DX = Handle
Output	AH = Error status AH=00H: O.K. AH=80H: Internal error, EMM possibly destroyed AH=81H: EMS hardware error AH=83H: Invalid handle AH=8CH: Mapping memory full AH=8DH: Mapping for handle already stored, not restored using function 48H

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information).

The handle used when calling this function must have been returned by a previous call to EMM function 43H.

This function is intended for use within a TSR program or by the operating system in a multitasking environment, but can be used by any program.

Interrupt 67H, Function 48H	LIM/EMS
Expanded memory: Restore mapping	

Restores mapping between the logical and physical pages saved by function 8H.

Input	AH = 48H DX = handle
Output	AH = Error status: AH=00H: O.K. AH=80H: Internal error, EMM possibly destroyed AH=81H: EMS hardware error AH=83H: Invalid handle AH=8EH: Mapping storage contains no entry for this handle

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information).

The handle used when calling this function must have been returned by a previous call to EMM function 43H.

Calling this function fails whenever the mapping for this handle has not been saved with function 47H, or the mapping has already been restored by a previous call to function 48H.

This function is intended for use within a TSR program or by the operating system in a multitasking environment, but can be used by any program.

Interrupt 67H, Function 49H	LIM/EMS
Expanded memory: Undocumented	
Interrupt 67H, Function 4AH	LIM/EMS
Expanded memory: Undocumented	
Interrupt 67H, Function 4BH	LIM/EMS
Expanded memory: Get number of handles	

Returns the number of memory blocks and the number of handles allocated by function 43H.

Input

AH = 4BH

Output

AH = 0: O.K.

BX = Number of allocated handles

AH > 0: Error

AH=80H: Internal error, EMM possibly destroyed

AH=81H: EMS hardware error

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information).

The number of allocated handles is not the same as the number of programs which are currently accessing the EMS memory. Each program can request an arbitrary number of EMS memory blocks/handles with function 4H.

Interrupt 67H, Function 4CH	LIM/EMS
Expanded memory: Get number of allocated pages	

Returns the number of pages which have been allocated to the specified handle.

Input

AH = 4CH

DX = Handle

Output

AH = 0: O.K.

BX = Number of allocated pages

AH > 0: Error

AH=80H: Internal error, EMM possibly destroyed

AH=81H: EMS hardware error

AH=83H: Invalid handle

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information).

The number of allocated pages must range from 1 to 512.

Interrupt 67H, Function 4DH**LIM/EMS**

Expanded memory: Get all handles

Loads the numbers of all active handles and the number of pages allocated to each into an array.

Input

AH = 4DH

ES = Segment address of array

DI = Offset address of array

Output

AH = 0: O.K.

BX = Number of allocated logical pages

AH > 0: Error

AH=80H: Internal error, EMM possibly destroyed

AH=81H: EMS hardware error

Remarks

Do not call this function unless you know that EMS memory and a corresponding EMM are installed (see Chapter 12 for more information).

If the function returns successfully, the memory area to which the ES:DI register pair points will contain two words for each active handle. The first word contains the handle itself and the second word contains the number of pages allocated to the handle. The number of these entries is returned in the BX register.

Since the EMM can manage a maximum of 256 handles, the array will never occupy more than 1024 bytes (1K).

Interrupt 67H, Function 4EH, Subfunction 00H**LIM/EMS (Version 3.2 and above)**

Expanded memory: Get page map

Gets page map (a map of logical and physical EMS pages).

Input AH = 4EH
 AL = 00H
 ES:DI = Pointer to empty array

Output AH = 0: O.K.
 ES:DI = Pointer to filled array
 AH >0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

This function requires no EMM handle (compare with function 48H).

Interrupt 67H, Function 4EH, Subfunction 01H	LIM/EMS (Version 3.2 and above)
Expanded memory: Set page map	

Sets the page map (a map of logical and physical EMS pages) to the status that existed before calling subfunction 00H.

Input AH = 4EH
 AL = 01H
 ES:DI = Pointer to page map array

Output AH = 0: O.K.
 AH >0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

This function requires no EMM handle (compare with function 48H).

Interrupt 67H, Function 4EH, Subfunction 02H	LIM/EMS (Version 3.2 and above)
Expanded memory: Swap page map	

Swaps the current page map with the previously stored page map.

Input AH = 4EH
 AL = 02H
 DS:SI = Pointer to the array containing the page map to be restored
 ES:DI = Pointer to the array containing the current page map

Output AH = 0: O.K.

AH >0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Before calling this function, the buffer containing the page map must have been previously loaded using subfunction 00H or subfunction 02H.

Subfunction 03H returns the size of the current page map array.

This function requires no EMM handle (compare with functions 47H and 48H).

Interrupt 67H, Function 4EH, Subfunction 03H

LIM/EMS (Version 3.2 and above)

Expanded memory: Get page map array size

Returns the amount of memory required for the page map.

Input AH = 4EH

AL = 03H

Output AH = 0: OK

AL=Page map array size in bytes

AH >0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Interrupt 67H, Function 4FH, Subfunction 00H

LIM/EMS (Version 4.0 and above)

Expanded memory: Save partial page map

Saves part of a page map.

Input AH = 4FH

AL = 00H

DS:SI = Pointer to map list

ES:DI = Pointer map state buffer

Output AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The map list must contain the number of pages in its first word, followed by a word containing the segment address of these pages. The size of this map list is therefore:

$2 + (\text{number_of_pages_to_be_stored} * 2)$ bytes

Subfunction 02H of this function determines the buffer size needed for saving the selected entries from the page map.

Interrupt 67H, Function 4FH, Subfunction 01H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Restore partial page map

Restores a page map previously saved using function 4FH, subfunction 00H.

Input

AH = 4FH

AL = 01H

DS:SI = Pointer to the buffer with the stored page table

Output

AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Before calling this function, the buffer must be saved using function 4FH, subfunction 00H.

Interrupt 67H, Function 4FH, Subfunction 02H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Get partial page map size info

Determines the size of the buffer required for saving part of a page map.

Input

AH = 4FH

AL = 02H

BX = Number of physical pages to be stored

Output

AH = 0: O.K.

AL=Array size in bytes

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Interrupt 67H, Function 50H, Subfunction 00H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Map multiple pages by number

Converts logical pages to physical pages, addressable through a numeric sequence.

Input

AH = 50H

AL = 00H

CX = Number of logical pages

DX = EMM handle by which pages are allocated

DS:SI = Pointer to buffer containing conversion information

Output

AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The buffer passed must contain two sequential words for every conversion. The first word specifies the logical page number, while the second word specifies the physical page number where the logical page will be reproduced.

If the logical page number is -1, the corresponding physical page is deleted and cannot be read or written.

Interrupt 67H, Function 50H, Subfunction 01H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Map multiple pages by address

Converts logical pages to physical pages, addressable through segment addresses.

Input

AH = 50H

AL = 01H

CX = Number of logical pages

DX = EMM handle by which pages are allocated

DS:SI = Pointer to buffer containing conversion information

Output

AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The buffer passed must contain two sequential words for every conversion. The first word specifies the logical page number, while the second word specifies the physical page number where the logical page will be reproduced.

If the logical page number is -1, the corresponding physical page is deleted and cannot be read or written.

Function 58H, subfunction 00H lets you determine the segment addresses of the available physical pages.

Interrupt 67H, Function 51H	LIM/EMS (Version 4.0 and above)
Expanded memory: Reallocate pages for handle	

Reallocates larger or smaller numbers of logical pages. This number belongs to a handle allocated by function 43H.

Input

AH = 48H

DX = EMM handle by which pages are allocated

BX = New number of pages

Output

AH = 0: O.K.

BX = Number of logical pages using this handle

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

You can reduce the number of pages assigned easily. However, increasing the number of pages allocated depends on the amount of free EMS memory available. Check the AH register for errors after every call to this function.

If this function reduces the number of allocated pages, a corresponding number of pages at the upper end of the pages allocated up to now are cut, destroying the contents of those pages.

The handle remains valid after the call, even if the number of pages is reduced to zero.

Interrupt 67H, Function 52H, Subfunction 00H	LIM/EMS (Version 4.0 and above)
Expanded memory: Get handle attribute	

Indicates whether EMS pages assigned to a handle are volatile (will not survive in memory after a warm boot) or volatile (will survive a warm boot).

Input

AH = 52H

AL = 00H

DX = EMM page handle

Output

AH = 0: O.K.

AL=Page attribute

0 : Volatile pages

1 : Non-volatile pages

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Interrupt 67H, Function 52H, Subfunction 01H**LIM/EMS (Version 4.0 and above)**

Expanded memory: Set handle attribute

If an EMS-card has a capability to protect EMS-pages against a Warmstart of the computer, the corresponding attribute can be determined with the help of this function.

Input

AH = 52H

AL = 01H

BL = Page attribute

0 : Volatile pages

1 : Non-volatile pages

DX = EMM page handle

Output

AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Only a few EMS cards can protect EMS pages from overwriting during a warm boot, and no EMS emulators can perform this task. Before calling this function, select subfunction 02H to determine the availability of non-volatile EMS pages.

Interrupt 67H, Function 52H, Subfunction 02H**LIM/EMS (Version 4.0 and above)**

Expanded memory: Get attribute capability

Determines whether the EMS card can offer protection to EMS pages from a warm boot. This function dictates whether function 52H, subfunctions 00H and 01H can be used.

Input

AH = 52H

AL = 02H

Output

AH = 0: O.K.

AL = 0 : Non-volatile pages not supported

AL = 1: Non-volatile pages supported

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Only a few EMS cards can protect EMS pages from overwriting during a warm boot, and no EMS emulators can perform this task. Call this function to determine the availability of non-volatile EMS pages.

Interrupt 67H, Function 53H, Subfunction 00H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Get handle name

Stores a handle name in the caller's buffer.

Input

AH = 53H

AL = 00H

DX = EMM page handle

ES:DI = Pointer to name buffer

Output

AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The name of a handle is always 8 characters in length. For this reason the buffer must have at least 8 bytes of storage capacity.

After function 43H allocates a handle, this new handle contains a 0 as its name. Before calling function 53H, subfunction 00H, call subfunction 01H to attach the name to a handle.

Interrupt 67H, Function 53H, Subfunction 01H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Set handle name

Gives a name to a previously allocated handle for access.

Input

AH = 53H

AL = 01H

DX = EMM page handle

ES:DI = Pointer to name buffer

Output

AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The name of a handle is always 8 characters in length. For this reason the buffer must have at least 8 bytes of storage capacity.

Interrupt 67H, Function 54H, Subfunction 00H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Get all handle names

Returns all handle names to a buffer in the caller.

Input

AH = 54H

AL = 00H

ES:DI = Pointer to name buffer

Output

AH = 0: O.K.

AL = Number of active handles

ES:DI = Pointer to filled in name buffer

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The buffer must offer 2,550 bytes of available memory (255 handles administered x 10 bytes for each handle entry). The first two bytes store the actual handle, and the remaining eight bytes contain the handle name.

Interrupt 67H, Function 54H, Subfunction 01H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Search for handle name

Searches for the handle name supplied by the caller.

Input

AH = 54H

AL = 01H

DS:SI = Pointer to name buffer

Output

AH = 0: O.K.

DX = EMM page handle

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Interrupt 67H, Function 54H, Subfunction 02H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Get total handles

Returns total number of handles.

Input AH = 54H

AL = 02H

Output AH = 0: O.K.

BX = Number of handle

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Interrupt 67H, Function 55H, Subfunction 00H

LIM/EMS (Version 4.0 and above)

Expanded memory: Map page by page number/jump

Maps the pages by page numbers, and jumps to one of the pages using a FAR jump.

Input AH = 55H

AL = 00H

DX = EMM page handle

DS:SI = Pointer to buffer

Output AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information). The buffer must have the following structure:

Addr.	Contents	Type
+00H	FAR pointer to the jump target	1 ptr
+05H	Number of pages to map before jump	1 byte
+06H	FAR pointer to map list	1 ptr
Length 9 bytes		

The structure of the map list is identical to the structure of this list during the construction of function 50H, subfunction 00H. The logical page number and its corresponding physical page are retained.

Interrupt 2FH, Code 55H, Subfunction 01H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Map page by page number/call

Maps the pages by page segments, and jumps to one of the pages using a FAR jump.

Input

AH = 55H

AL = 01H

DX = EMM page handle

DS:SI = Pointer to buffer

Output

AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The buffer must have the following format:

Addr.	Contents	Type
+00H	FAR pointer to the jump target	1 ptr
+05H	Number of pages to map before jump	1 byte
+06H	FAR pointer to map list	1 ptr
Length 9 bytes		

The format of the map list is identical to the format of this list during the execution of function 50H, subfunction 00H. The logical page number and its corresponding physical page are retained.

Interrupt 67H, Function 56H, Subfunction 01H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Map page by page segment/call

Maps pages by page segments and calls a program through a FAR call. After this routine ends, a second page map occurs, and interrupt function control returns to the caller.

Input

AH = 56H

AL = 01H

DX = EMM page handle

DS:SI = Pointer to the buffer

Output

AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The buffer must have the following format:

Addr.	Contents	Type
+00H	FAR pointer to the call's target	1 ptr
+05H	Number of pages to be mapped before call	1 byte
+06H	FAR pointer to list of pages to be mapped before call	1 ptr
+0AH	Number of pages to map before returning	1 byte
+0BH	FAR pointer to list of pages to be mapped before return	1 ptr
Length: 14 byte		

The format of the map list is identical to the format of this list during the execution of function 50H, subfunction 00H. The logical page number and its corresponding physical page are retained.

Call subfunction 02H to ensure that enough memory is available before calling this function.

Interrupt 67H, Function 56H, Subfunction 02H	LIM/EMS (Version 4.0 and above)
Expanded memory: Get stack space for map page/call	

Returns the memory space required by subfunctions 00H and 01H.

Input	AH = 56H
	AL = 02H
Output	AH = 0: O.K.
	BX = Space required
	AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Interrupt 67H, Function 57H, Subfunction 00H	LIM/EMS (Version 4.0 and above)
Expanded memory: Move memory region	

Copies memory regions between expanded memory and conventional memory, or within either set of memory.

Input	AH = 57H
	AL = 00H
	DS:SI = Pointer to buffer

Output AH = 0: O.K.

 AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The buffer must have the following format:

Addr.	Contents	Type
+00H	Region length in bytes	1 dword
+04H	Source memory type (0 =conventional, 1 =EMS)	1 byte
+05H	Source memory handle (EMS only)	1 word
+07H	Source memory offset	1 word
+09H	Source memory segment address (conv. memory) or logical page number (EMS)	1 word
+0BH	Target memory type (0 =conventional, 1 =EMS)	1 byte
+0CH	Target memory handle (EMS only)	1 word
+0EH	Target memory offset	1 word
+10H	Target memory segment address (conv. memory) or logical page number (EMS)	1 word
Length: 18 bytes		

Interrupt 2FH, Code 57H, Subfunction 01H

LIM/EMS (Version 4.0 and above)

Expanded memory: Exchange memory regions

This function operates similar to the subfunction 00H. It does not copy one storage area into another, but swaps the content of the indicated storage areas.

Input AH = 57H

 AL = 01H

 DS:SI = Pointer to buffer

Output AH = 0: O.K.

 AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The buffer must have the following format:

Addr.	Contents	Type
+00H	Region length in bytes	1 dword
+04H	Source memory type (0 = onventional, 1 =EMS)	1 byte
+05H	Source memory handle (EMS only)	1 word
+07H	Source memory offset	1 word
+09H	Source memory segment address (conv. memory) or logical page number (EMS)	1 word
+0BH	Target memory type (0 =conventional, 1 =EMS)	1 byte
+0CH	Target memory handle (EMS only)	1 word
+0EH	Target memory offset	1 word
+10H	Target memory segment address (conv. memory) or logical page number (EMS)	1 word
Length: 18 bytes		

This function can swap up to 1 megabyte of memory.

If EMS pages are participating in this swap and the size specified exceeds the EMS page, other EMS pages will be used.

The source and destination regions may not overlap.

Interrupt 67H, Function 58H, Subfunction 00H	LIM/EMS (Version 4.0 and above)
Expanded memory: Get addresses of mappable pages	

Returns addresses of all physical EMS pages and related page numbers.

Input

AH = 58H

AL = 00H

ES:DI = Pointer to buffer

Output

AH = 0: O.K.

CX=Number of entries

ES:DI=Pointer to filled in buffer

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Subfunction 01H helps determine the size of the buffer required.

EMM passes the desired information into the buffer in two word entries. The first word contains the segment address of an EMS page, while the second word indicates the page number that applies to the page.

Interrupt 67H, Function 58H, Subfunction 01H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Get number of mappable pages

Determines the number of physical EMS pages, and calculates the size of the buffer required by subfunction 00H.

Input

AH = 58H

AL = 01H

Output

AH = 0: O.K.

CX=Number of pages

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The size of the buffer required is the result of multiplying the number of pages (the contents of the CX register) by four.

Interrupt 67H, Function 59H, Subfunction 00H**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: Get hardware configuration

Determines EMS hardware configuration. This function only applies to an operating system equipped with EMS.

Input

AH = 59H

AL = 00H

ES:DI = Pointer to buffer

Output

AH = 0: O.K.

AL=Page map size in bytes

ES:DI=Pointer to filled in buffer

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

The buffer must be a minimum size of 10 bytes. It contains the following information after the function call:

Addr.	Contents	Type
+00H	Raw EMS page size in paragraphs	1 word
+02H	Alternate EMS register sets	1 word
+04H	Mapping context save area size in bytes	1 word
+06H	Number of assignable register sets	1 word
+08H	DMA operating mode (0 = DMA with alt register sets 1 = One DMA register set)	1 word
Length: 10 bytes		

Interrupt 67H, Function 59H, Subfunction 01H	LIM/EMS (Version 4.0 and above)
Expanded memory: Get number of raw pages	

Determines the total number of raw (non-standard EMS) pages, and indicates which of those pages are available. A raw page has a size other than the default 16K.

Input AH = 59H

AL = 01H

Output AH = 0: O.K.

DX=Total number of raw pages

BX=Number of raw pages not allocated

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information). Not all EMS cards support raw pages. This may result in a value of zero returned in the DX register.

Interrupt 67H, Function 5AH, Subfunction 00H	LIM/EMS (Version 4.0 and above)
Expanded memory: Allocate handle & standard pages	

Allocates standard EMS pages. Unlike function 43H, this function can allocate zero pages without error.

Input AH = 5AH

AL = 0H

BX = Number of standard pages

Output AH = 0: O.K.

DX=EMM page handle

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Interrupt 67H, Function 5AH, Subfunction 01H	LIM/EMS (Version 4.0 and above)
Expanded memory: Allocate handle & raw pages	

Allocates raw pages. A raw page has a size other than the default 16K.

Input

AH = 5AH

AL = 01H

BX = Number of pages to be allocated

Output

AH = 0: O.K.

DX=EMM page handle

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

Not all EMS cards support raw pages. Test the function status in the AH register.

This function can also allocate zero pages.

Interrupt 67H, Function 5BH	LIM/EMS (Version 4.0 and above)
Expanded memory: Alternate map and DMA services	

Function 5BH contains nine subfunctions used exclusively in operating system development. None of these subfunctions are used in normal PC software development.

Interrupt 67H, Function 5CH	LIM/EMS (Version 4.0 and above)
Expanded memory: Prepare EMM for warm boot	

Prepares the Expanded Memory Manager for an imminent warm boot of the computer. This gives the EMM an opportunity to store internal data and prevent the loss of non-volatile pages when the warm boot occurs.

Input

AH = 5CH

Output

AH = 0: O.K.

AH > 0: Error

Remarks

Do not call this function unless you know that EMS memory and a corresponding version of EMM are installed (see Chapter 12 for more information).

See also function 52H for more information.

Interrupt 67H, Function 5DH**LIM/EMS** (*Version 4.0 and above*)

Expanded memory: EMM operating system services

Function 5BH contains three subfunctions used exclusively in operating system development. None of these subfunctions are used in normal PC software development.

XMS Functions

The XMS (eXtended Memory Specification) was created by Microsoft in cooperation with several other corporations, as a supplement to the EMS standard. This specification coordinates access to extended memory, the memory which lies beyond the 1 megabyte limit of conventional memory.

Interrupt 2FH must recognize an XMS driver, and the XMS handler's address must be specified, before calling any of these functions. XMS functions are called through a FAR CALL instruction, instead of the special interrupt used by other function interfaces. See Chapter 12 for more information.

Error codes are placed in the BL register. The following error codes can occur during XMS function calls:

80H	Function not implemented	A4H	Source offset is invalid
81H	VDISK device driver was detected	A5H	Destination handle is invalid
82H	A20 error	A6H	Destination offset is invalid
8EH	General driver error	A7H	Length is invalid
8FH	Unrecoverable driver error	A8H	Overlap in move request is invalid
90H	HMA does not exist	A9H	Parity error detected
91H	HMA is already in use	AAH	Block not locked
92H	DX is less than /HMAMIN = parameter	ABH	Block locked
93H	HMA is not allocated	ACH	UMB count overflowed
94H	A20 line is still enabled	ADH	Lock failed
A0H	All extended memory is allocated	B0H	Smaller UMB is available
A1H	EMM handles are exhausted	B1H	No UMBs are available
A2H	Handle is invalid	B2H	UMB segment number is invalid
A3H	Source handle is invalid		

Function 00H	XMS
Determine XMS version number	

Returns the XMS driver's version number and the driver's internal revision number.

Input	AH = 00H
Output	AX = XMS version number BX = Internal revision number

DX = HMA status
 0: No HMA
 1: HMA available

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

Function 01H	XMS
Allocate High Memory Area (HMA)	

Allocate High Memory Area (HMA)

Reserves all or part of the HMA for program use.

Input AH = 01H
 DX = Requested HMA space in bytes
 FFFFH: Amount needed for application program
 < FFFFH: Actual amount needed for operating system or driver

Output AX = 0001H: Operation completed
 AX = 0000H: Error
 BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

Address line A20 must be freed using function 03H or function 05H, before HMA access can occur.

This function can fail if a TSR doesn't request the entire HMA, and if the amount of memory requested is less than the value passed by the /HMAMIN parameter during driver installation. These factors give a TSR, which may require only a few kilobytes, exclusive access rights to the HMA, while prohibiting access by another program requiring more memory.

If the HMA should not remain in possession of the program beyond its termination, it must be freed before the end of program execution by function 02H.

Function 02H	XMS
Free High Memory Area (HMA)	

Releases the HMA previously allocated by function 01H.

Input AH = 02H

Output AX = 0001H: Operation completed
 AX = 0000H: Error

BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

A previously allocated HMA remains allocated even after program execution ends. This function must be called before the HMA can be allocated by other subsequent program calls.

Calling function 02H destroys the HMA's current contents.

Function 03H

XMS

Globally enable address line A20

Globally enables address line A20, permitting direct HMA access when the processor is in real mode.

Input

AH = 03H

Output

AX = 0001H: Operation completed

AX = 0000H: Error

BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

This function must be called before or after allocating the HMA. This ensures HMA access in real mode through segment address FFFFH.

Function 04H should be used to disable address line A20 before program execution ends. This prevents a possible segment overflow in subsequent program calls.

The process of enabling address line A20 can take much time. We recommend that you call function 03H only when necessary.

Function 04H

XMS

Globally disable address line A20

Globally disables address line A20, prohibiting direct HMA access when the processor is in real mode.

Input

AH = 04H

Output

AX = 0001H: Operation completed

AX = 0000H: Error

BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

The process of disabling address line A20 can take much time. We recommend that you call function 04H only when necessary.

Function 05H	XMS
Locally enable address line A20	

Locally enables address line A20. Calls to functions 05H and 06H are recorded internally with the help of a counter, which increments with every call to function 05H and decrements with every call to function 06H. This line can be enabled only if it is currently disabled.

Input	AH = 05
Output	AX = 0001H: Operation completed
	AX = 0000H: Error
	BL = Error code (see below)

Remarks

This function must be called before or after allocating the HMA.

Function 06H should be used to disable address line A20 before program execution ends. This prevents a possible segment overflow in subsequent program calls.

The process of enabling address line A20 can take much time. We recommend that you call function 05H only when necessary.

Function 06H	XMS
Locally disable address line A20	

Locally disable address line A20, if line A20 was previously enabled by function 05H.

Input	AH = 06
Output	AX = 0001H: Operation completed
	AX = 0000H: Error
	BL = Error code (see below)

Remarks

Calls to functions 05H and 06H are recorded internally with the help of a counter, which increments with every call to function 05H and decrements with every call to function 06H. Only after reaching the value 0 is the address line actually disabled. Every call to function 06H must be preceded by a call to function 05H.

Function 06H should be used to disable address line A20 before program execution ends. This prevents a possible segment overflow in subsequent program calls.

The process of enabling address line A20 can take much time. We recommend that you call function 05H only when necessary.

Function 07H	XMS
Query status of address line A20	

Determines whether address line A20 has been enabled by functions 03H or 05H.

Input AH = 07H

Output AX = 0001H: Address line A20 enabled

AX = 0000H: Address line A20 disabled

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

This query is hardware-controlled, and takes less time than the process of enabling or disabling address line A20.

Function 08H	XMS
Query free extended memory	

Returns the total amount of free extended memory, as well as the size of the largest free block of extended memory.

Input AH = 08H

Output AX = Size of the largest free block of extended memory in kilobytes

DX = Total free extended memory in kilobytes

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

The HMA is excluded from this function, even if no HMA has been allocated. This results in a total 64K too large, because extended memory allocation always starts after the HMA. See Chapter 12 for more information.

Function 09H	XMS
Allocate Extended Memory Block (EMB)	

Allocates an EMB for program use.

Input AH = 09H

DX = Size of the requested block in kilobytes

Output AX = 0001H: Operation completed

AX = 0000H: Error

BL = Error code (see below)

DX = Handle for additional EMB access, if additional access is possible

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

The returned handle accesses the EMB during all subsequent calls, and must therefore be stored in the program. If the program loses the handle, the user can only free the EMB by resetting the computer.

The XMS can only honor an EMB allocation if a handle is free, and if a large enough memory block is available. Since the number of handles is limited (usually 32), large EMBs should be allocated to avoid calling the function too often, and to minimize handle calls.

The allocated EMB must be freed using 0AH before program execution ends, or the EMB will be lost for passing to subsequent applications.

Function 0AH	XMS
Free allocated Extended Memory Block (EMB)	

Frees an Extended Memory Block (EMB) previously allocated using function 09H.

Input	AH = 0AH
	DX = Handle
Output	AX = 0001H: Operation completed
	AX = 0000H: Error
	BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

Calling function 0AH destroys the EMB's current contents and invalidates the handle.

Function 0BH	XMS
Move Extended Memory Block (EMB)	

Transfers memory between conventional RAM and extended memory, or copies blocks within conventional RAM or extended memory.

Input	AH = 0BH
	DS:SI = Pointer to the following structure which determines the memory area to be copied and its destination.
Output	AX = 0001H: Operation completed
	AX = 0000H: Error
	BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

A handle value of 0 indicates access to conventional RAM. The segment and offset addresses specify the beginning of the memory block.

When addressing an EMB, the handle returned after allocating the EMB using function 09H must be indicated. The offset address then represents the offset relative to the start of the block.

The handles specified cannot be handles whose corresponding EMB was locked using function 0CH.

If the indicated blocks overlap, the source block must precede the destination block, or the function may fail.

The source block copies faster if both blocks begin at even numbered addresses in an AT, or if both blocks begin at addresses divisible by four in an 80386.

The Extended Memory Move Structure		
Addr	Content	Type
+00H	Block length in bytes (must be an even number)	1 DWORD
+04H	Handle of the source Block	1 WORD
+06H	Source block offset, where copying starts	1 DWORD
+0AH	Handle of the destination Block	1 WORD
+0CH	Destination block offset, where copying starts	1 DWORD
Length: 16 bytes		

Function 0CH	XMS
Lock Extended Memory Block (EMB)	

Locks an Extended Memory Block (EMB) to a specific memory location. The XMM moves EMBs to different locations after the release of an EMB, thus preventing memory gaps. Function 0CH ensures that the specified EMB remains in the same location in memory.

Input

AH = 0CH

DX = EMB handle

Output

AX = 0001H: Operation completed

AX = 0000H: Error

BL = Error code (see below)

DX:BX = Linear 32-bit address of the EMB in memory

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

The function call returns the EMB's address. This address is valid until the EMB is freed.

Release locked blocks as soon as possible using function 0DH. This minimizes any hindering of the XMM's tasks.

Calls to functions 0CH and 0DH are recorded internally with the help of a counter, which increments with every call to function 0CH and decrements with every call to function 0DH. Only after reaching the value 0 can the corresponding EMB be unlocked.

Function 0DH	XMS
Unlock Extended Memory Block (EMB)	

Unlocks an Extended Memory Block (EMB) previously locked using function 0DH. The XMM moves EMBs to different locations after the release of an EMB, thus preventing memory gaps. Function 0DH ensures that the specified EMB is also available to the XMM.

Input	AH = 0DH
	DX = EMB handle
Output	AX = 0001H: Operation completed
	AX = 0000H: Error
	BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

This function invalidates the EMB address returned during the call of function 0CH.

Calls to functions 0CH and 0DH are recorded internally with the help of a counter, which increments with every call to function 0CH and decrements with every call to function 0DH. Only after reaching the value 0 can the corresponding EMB be unlocked.

Function 0EH	XMS
Get (EMB) handle information	

Provides information about an EMB. This includes the block size, its block counter and the number of handles free.

Input	AH = 0EH
	DX = EMB handle
Output	AX = 0001H: Operation completed
	BH = Block counter
	BL = Number of free EMB handles
	DX = EMB length in kilobytes
	DX = 0000H: Error
	BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

Calls to functions 0CH and 0DH are recorded internally with the help of a block counter, which increments with every call to function 0CH and decrements with every call to function 0DH. A value unequal to zero indicates the number of calls which must be made to 0DH before the EMB can be unlocked.

Function 0FH	XMS
Resize Extended Memory Block (EMB)	

Allows the enlargement or reduction of the size of an EMB.

Input	AH = 0FH
	BX = New size in kilobytes
	DX = EMB handle
Output	AX = 0001H: Operation completed
	AX = 0000H: Error
	BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

The indicated EMB must not be locked.

If you reduce the size of the EMB, the contents of the upper end of the EMB are lost.

Function 10H	XMS
Allocate Upper Memory Block (UMB)	

Allocates an upper memory block in the RAM existing between the 640K limit and the beginning of extended memory.

Input	AH = 10H
	DX = Size of the requested memory blocks in paragraphs
Output	AX = 0001H: Operation completed
	BX = Segment address of the UMB
	BX = 0000H: Error
	BL = Error code (see below)
	DX = Maximum size of an allocatable UMB in paragraphs

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

This function is extremely hardware dependent and not implemented in all XMS drivers. After its call the error status must be checked to ensure that a UMB actually was allocated.

Direct UMB access is possible in real mode with the returned segment address. During this access, the offset address derived from the conversion of block length may not be exceeded.

This function can also be used to determine the largest available UMB, by using an unrealistic value (for example FFFFH) for the size of the requested block. No UMB is allocated, but the function returns the length of the largest available UMB.

Function 11H	XMS
Free allocated Upper Memory Block (UMB)	

Frees a UMB previously allocated using function 10H, making the UMB available to other programs.

Input

AH = 11H

DX = UMB segment address

Output

AX = 0001H: Operation completed

AX = 0000H: Error

BL = Error code (see below)

Remarks

Interrupt 2FH must recognize an XMS driver, and the XMS driver jump location must be specified, before calling this function.

After calling function 11H, the UMB's contents are lost. No memory access can be made through the block's segment address.

All allocated UMBs should be released using this function before program execution ends. Otherwise, it may not be possible to pass the UMBs to subsequent programs.

Mouse Driver Functions

Microsoft has supported its mouse with a software-interface in the form of the MOUSE.SYS device driver or the MOUSE.COM program. This interface has been established and is imitated by all other mouse manufacturers. This insures full compatibility with the Microsoft mouse. The various functions of the mouse-interface are called through Interrupt 33h, where a 16 bit value is used as function number, which must be passed in the AX-Register.

Version 8.0 of the mouse driver supports 53 different functions, with function numbers from 0000h to 00034h. However, the functions 11h and 12h, as well as 002Eh are undocumented and therefore are only used inside the mouse driver.

The Microsoft mouse-driver has been subjected to many revisions and enhancements. Therefore numerous variants exist between the version 1.0 and 8.0. Versions with a version number smaller than 6.26 are rarely still in circulation, because they do not work with the latest video cards.

Unfortunately, Microsoft reveals the date of the introduction of the various mouse driver functions only since the function 25h, which was introduced with version 6.26. Many of preceding functions existed since version 1.0, but many were added during the course of development between version 1.0 and 6.26. Since more accurate information is lacking, it will be assumed in the framework of this reference, that these functions have been available since version 1.0. Even if this is not true in a particular case, you can assume that these functions are supported by all mouse-drivers which are in use.

Most mouse functions operate with the AX, BX, CX and DX, registers to pass information from the caller and to return function results. Only in exceptional cases are the ES, SI and DS registers used for the function call, usually when the address of buffers must be passed.

Basically only the content of the register changes for the return of the function results. The content of all other registers remains unchanged. For most functions, the AX-Register is used for the return of the function status, which in case of error indicates the failure of the operation.

Interrupt 33H, Function 00H	Mouse
Resets (initializes) the mouse driver	

Resets (initializes) the mouse driver.

Input	AX = 00H
Output	AX = Mouse installation status
	AX=FFFFH: Mouse driver installed
	AX=0000H: Error, no mouse driver installed
	BX = Number of mouse buttons

Remarks

The reset process executes the following tasks:

Moves the mouse cursor; to the center of the screen and clears the cursor from the screen. When enabled, the default cursor appears as an inverse video square. The representation is always in display page 0, independent of the current display mode. The entire screen area becomes the total range of mouse movement.

Installs the event handler is installed by a program (default is disabled).

Installs light pen emulation (default is disabled).

Specifies mouse cursor's speed. Default relative speed is 8 mickeys per 8 horizontal pixels and 16 mickeys per 16 vertical pixels.

Specifies maximum mouse speed (default is 64 mickeys per second).

Interrupt 33H, Function 01H	Mouse
Display mouse cursor	

Displays the mouse cursor on the screen. This cursor follows any movement the user makes with the mouse device.

Input AX = 01H

Output No output

Remarks

This function increments an internal counter which determines whether the mouse cursor should be displayed on the screen. When the mouse driver is initialized using function 00H, this cursor contains the value -1 (i.e., the mouse cursor does not appear). If this counter contains the value 0 after calling function 01H, the mouse cursor appears on the screen.

The mouse driver follows the mouse movement even when the mouse cursor is not displayed on the screen. After calling this function, the mouse cursor may not appear at the same location as it was when the cursor was previously removed by calling function 00H or function 02H.

Interrupt 33H, Function 02H	Mouse
Hide mouse cursor	

Removes the mouse cursor from the screen.

Input AX = 02H

Output No output

Remarks

This function decrements an internal counter which determines whether the mouse cursor should appear on the screen. If the counter contains the value 0, the mouse cursor is displayed on the screen, while the value -1 removes the mouse cursor from the screen.

The mouse driver follows the mouse movement even when the mouse cursor is not displayed on the screen.

After calling this function, the mouse cursor may not appear at the same location as it was when the cursor was previously removed by calling function 00H or function 02H.

Interrupt 33H, Function 03H	Mouse
Get cursor position/button status	

Returns the current position of the mouse cursor and the current status of the mouse buttons.

Input AX = 03H

Output BX = Mouse button status

Bit 0=1: Left mouse button activated

Bit 1=1: Right mouse button activated

Bit 2=1: Center mouse button activated

Bits 3-15: Unused

CX = X coordinate (horizontal mouse position)

DX = Y coordinate (vertical mouse position)

Remarks

The coordinates returned in the CX and DX registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen.

If the mouse is equipped with only two mouse buttons, the information about the central mouse button does not have significance.

Interrupt 33H, Function 04H	Mouse
Move mouse cursor	

Moves the active mouse cursor to a certain position on the screen.

Input AX = 04H

CX = X coordinate (horizontal mouse position)

DX = Y coordinate (vertical mouse position)

Output No output

Remarks

The coordinates returned in the CX and DX registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen.

If the position indicated is outside the range of movement specified by functions 07H and 08H, the function adjusts coordinates so that the mouse cursor remains within this range of movement.

The mouse cursor moves to the new position, even if the mouse is not currently visible. Once re-enabled, the mouse cursor appears at this new position.

Interrupt 33H, Function 05H	Mouse
Determine number of times mouse button was activated	

Informs the calling program of how often a mouse button has been pressed since the last call of function 05H. Function 05H also informs the calling program of the cursor's location on the screen when the button was last activated.

Input

AX = 05H

BX = Mouse button activated

BX=0: Left mouse button

BX=1: Right mouse button

BX=2: Center mouse button

Output

BX = Status of all mouse buttons:

Bit 0=1: Left mouse button activated

Bit 1=1: Right mouse button activated

Bit 2=1: Center mouse button activated

Bits 3-15: Unused

BX = Mouse buttons activated since last function call

CX = Horizontal mouse position during the last activation

DX = Vertical mouse position during the last activation

Remarks

The coordinates returned in the CX and DX registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen. The activation counter for the mouse button addressed is reset to 0 when this function is called.

Interrupt 33H, Function 06H	Mouse
Determine number of times mouse button was released	

Informs the calling program of how often a mouse button has been released since the last call of function 06H. Function 06H also informs the calling program of the cursor's location on the screen when the button was last activated.

Input

AX = 06H

BX = mouse button addressed

BX=0: Left mouse button

BX=1: Right mouse button

BX=2: Center mouse button

Output	BX = Status of all mouse buttons
	Bit 0=1: Left mouse button activated
	Bit 1=1: Right mouse button activated
	Bit 2=1: Center mouse button activated
	Bits 3-15: Unused
	BX = Mouse buttons activated since last function call
	CX = Horizontal mouse position during the last activation
	DX = Vertical mouse position during the last activation

Remarks

The coordinates returned in the CX and DX registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen.

The activation counter for the mouse button addressed is reset to 0 when this function is called.

Interrupt 33H, Function 07H	Mouse
Set horizontal range of movement	

Defines the horizontal range of movement for the mouse cursor. Once set, the user cannot move the mouse cursor out of this range.

Input	AX = 07H
	CX = Minimal horizontal cursor position
	DX = Maximum horizontal cursor position

Output	No output
---------------	-----------

Remarks

The coordinates passed in the CX and DX registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen.

If the mouse cursor is outside of this range when function 07H is called, the mouse driver automatically moves the mouse cursor within the limits of the range of movement. If the value in the DX register is less than the value in the CX registers, the two parameters are exchanged.

Interrupt 33H, Function 08H	Mouse
Set vertical range of movement	

Defines the vertical movement range for the mouse cursor. When set, the user cannot move the mouse cursor out of this range.

Input	AX = 08H
	CX = Minimum vertical cursor position
	DX = Maximum vertical cursor position

Output No output

Remarks

The coordinates passed in the CX and DX registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen.

If the mouse cursor is outside of this range when function 07H is called, the mouse driver automatically moves the mouse cursor within the limits of the range of movement.

If the value in the DX register is less than the value in the CX registers, the two parameters are exchanged.

Interrupt 33H, Function 09H	Mouse
Set mouse cursor (graphic mode)	

Defines the appearance of the mouse cursor in graphic mode, as well as the ..bitfield; which compensates for the pixels around the mouse cursor.

Input

AX = 09H

BX = Cursor width starting at left border of bitfield

CX = Cursor height starting at top border of bitfield

ES = Segment address of bitfield

DX = Offset address of bitfield

Output No output

Remarks

The bitfield consists of 64 bytes, of which the first 32 are an AND comparison, and the remaining 32 are an OR combination. Both sets of bytes are based upon the current pixel pattern.

Interrupt 33H, Function 0AH	Mouse
Set mouse cursor (text mode)	

Defines the bitmask which specifies the appearance of the mouse cursor in text mode.

Input

AX = 0AH

BX = Cursor type

BX=0: Software cursor

BX=1: Hardware cursor

CX = AND mask (software cursor) or starting line (hardware cursor)

DX = XOR mask (software cursor) or ending line (hardware cursor)

Output No output

Remarks

If the software cursor is selected, the code of the character beneath the mouse cursor and its attribute byte are combined logically with the mask in the CX register through a binary AND, and then with the value in the DX register through an exclusive OR (XOR). The attribute byte is combined with the most significant byte (CH and DH). The character code is combined with the least significant byte (CL and DL).

The hardware cursor is the same shape as the normal text mode cursor. Monochrome mode values for the starting and ending lines range from 0 to 13. Color mode values for the starting and ending lines range from 0 to 7.

Interrupt 33H, Function 0BH	Mouse
Determine movement values	

Determines the distance between the current mouse position and the mouse position during the last call of function 0BH.

Input	AX = 0BH
Output	CX = Horizontal distance from last point in mickeys DX = Vertical distance from last point in mickeys

Remarks

These values must be interpreted as signed numbers. Positive values indicate movement toward the bottom or right border of the screen, while negative values indicate movement toward the top or left border of the screen.

These values are given in mickeys.(1 mickey=1/200 inch) rather than in pixels.

Interrupt 33H, Function 0CH	Mouse
Set event handler	

Sets the address of an event handler called by the mouse driver when a particular mouse event occurs.

Input	AX = 0CH
	CX = Events which trigger the call of the event handler (event mask)
	Bit 0: Mouse movement
	Bit 1: Left mouse button activated
	Bit 2: Left mouse button released
	Bit 3: Right mouse button activated
	Bit 4: Right mouse button released
	Bit 5: Center mouse button activated
	Bit 6: Center mouse button released
	Bits 7-15: Unused
	ES = Segment address of handler
	DX = Offset address of handler

Output No output

Remarks

The event handler is called by the mouse driver through a FAR call assembler instruction, and therefore must be terminated with a FAR RET instruction. None of the various processor registers may be returned to the caller with a changed content.

The mouse driver passes the following information to the event handler through the processor registers during the call:

- AX = Event mask. The bits correspond to the various events as indicated in the CX register during the installation of the event handler. In addition, other bits can be set, since the value reflects the current status of the mouse driver, and is not limited to the selected events.
- BX = Mouse button status:
 - Bit 0 = Left mouse button activated
 - Bit 1 = Right mouse button activated
 - Bit 2 = Center mouse button activated
- CX = Horizontal mouse position.
- DX = Vertical mouse position.
- SI = Length of last horizontal mouse movement.
- DI = Length of the last vertical mouse movement.
- DS = Data segment of the mouse driver.

The coordinates returned in the CX and DX registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen.

The values in the SI and DI registers refer to mickeys (one mickey = 1/200 inch).

These mickey values must be interpreted as signed numbers. Positive values indicate movement toward the bottom or right border of the screen, while negative values indicate movement toward the top or left border of the screen.

Interrupt 33H, Function 0DH	Mouse
Enable light pen emulation	

Enables emulation of the light pen, and simulates a light pen which if none is present.

Input AX = 0DH

Output No output

Remarks

Light pen emulation only makes sense when used with an application which supports the light pen, or makes light pen reading routines available (e.g., the PEN command in PC-BASIC).

The light pen and mouse are closely related in programming: The position of the mouse cursor is directly related to the light pen's position on the screen, and pressing the left and right mouse button has the same result as pressing the button on the light pen.

Interrupt 33H, Function 0EH	Mouse
Disable light pen emulation	

Disables the light pen emulation enabled by a previous call to function 0DH.

Input AX = 0EH

Output No output

Remarks

Light pen emulation only makes sense when used with an application which supports the light pen, or makes light pen reading routines available (e.g., the PEN command in PC-BASIC).

The light pen and mouse are closely related in programming: The position of the mouse cursor is directly related to the light pen's position on the screen, and pressing the left and right mouse button has the same result as pressing the button on the light pen.

Interrupt 33H, Function 0FH	Mouse
Set cursor speed	

Defines the relationship between mickeys and screen pixels. This specifies the sensitivity of the mouse and the speed at which the mouse cursor moves across the screen.

Input AX = 0FH

 CX = Number of horizontal mickeys

 DX = Number of vertical mickeys

Output No output

Remarks

Values in the CX and DX registers can range from 1 to 32767.

The default setting is 8 horizontal mickeys and 16 vertical mickeys. This causes the mouse cursor to move twice as fast horizontally as it moves vertically. Calling function 00H (Reset mouse driver) changes any previously set values to the default values.

Interrupt 33H, Function 10H	Mouse
Exclusion area	

Designates any area of the screen as an exclusion area. The mouse cursor disappears if moved into the exclusion area.

Input AX = 10H

 CX = X-coordinate, upper left corner of exclusion area

 DX = Y-coordinate, upper left corner of exclusion area

 SI = X-coordinate, lower right corner of exclusion area

 DI = Y-coordinate, lower right corner of exclusion area

Output No output

Remarks

The coordinates passed in the CX, DX, DI and SI registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen.

Calling function 00H (Reset mouse driver) or function 01H (Display mouse cursor) deletes the exclusion area coordinates.

Interrupt 33H, Function 11H	Mouse
Undocumented	

This undocumented function is used exclusively by the mouse driver, and cannot be called from a program.

Interrupt 33H, Function 12H	Mouse
Undocumented	

This undocumented function is used exclusively by the mouse driver, and cannot be called from a program.

Interrupt 33H, Function 13H	Mouse
Set maximum for mouse speed doubling	

Sets the maximum limit for doubling mouse speed. If the speed of the mouse movement exceeds a certain limit, the mouse driver doubles the mouse cursor speed by doubling the movement's relationship between points and mickeys.

Input AX = 13H
DX = Limit in mickeys per second

Output No output

Remarks

1 mickey=1/200 inches.

To prevent doubling of the mouse speed, the limit can be set higher.

Speeds in excess of 5,000 mickeys per second cannot be achieved by practical means.

Interrupt 33H, Function 14H	Mouse
Exchange event handlers	

Installs a new event handler for certain mouse events, but also retains the address of the old event handler.

Input AX = 14H
CX = Events which should trigger event handler call
Bit 0: Mouse movement
Bit 1: Left mouse button activated
Bit 2: Left mouse button released

Bit 3: Right mouse button activated

Bit 4: Right mouse button released

Bit 5: Center mouse button activated

Bit 6: Center mouse button released

Bit 7-15: Unused

ES = Segment address of new event handler

DX = Offset address of new event handler

Output

CX = Event mask of the previously installed event handler

ES = Segment address of previously installed event handler

DX = Offset address of previously installed event handler

Remarks

The event handler is called by the mouse driver through a FAR call assembler instruction, and therefore must be terminated with a FAR RET instruction. None of the various processor registers may be returned to the caller with a changed content.

The mouse driver passes the following information to the event handler through the processor registers during the call:

AX = Event mask. The bits correspond to the various events as indicated in the CX register during the installation of the event handler. In addition, other bits can be set, since the value reflects the current status of the mouse driver, and is not limited to the selected events.

BX = Mouse button status:

Bit 0 = Left mouse button activated

Bit 1 = Right mouse button activated

Bit 2 = Center mouse button activated

CX = Horizontal mouse position.

DX = Vertical mouse position.

SI = length of last horizontal mouse movement.

DI = Length of the last vertical mouse movement.

DS = Data segment of the mouse driver.

The coordinates returned in the CX and DX registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen.

The values in the SI and DI registers refer to mickeys (one mickey = 1/200 inch).

These mickey values must be interpreted as signed numbers. Positive values indicate movement toward the bottom or right border of the screen, while negative values indicate movement toward the top or left border of the screen.

Interrupt 33H, Function 15H	Mouse
Determine mouse status buffer size	

Returns the size of the mouse status buffer, in which a program can store the complete status of the mouse driver.

Input AX = 15H

Output BX = Mouse status buffer size in bytes

Remarks

Function 16H (Store mouse status) stores the mouse status in the buffer.

Interrupt 33H, Function 16H	Mouse
Store mouse status	

Stores mouse status information in a buffer.

Input AX = 16H

 ES = Segment address of mouse status buffer

 DX = Offset address of mouse status buffer

Output No output

Remarks

The caller is responsible for creating a buffer large enough to contain all the status information. Before calling this function, call function 15H (Determine mouse status buffer size) to determine the size of the mouse status buffer.

This function works well when called before executing a program using the EXEC function. This allows the mouse status to be saved in memory, then restored from within the called program.

Interrupt 33H, Function 17H	Mouse
Restore mouse status	

Reads all mouse parameters from a buffer stored by function 16H.

Input AX = 17H

 ES = Segment address of mouse status buffer

 DX = Offset address of mouse status buffer

Output No output

Interrupt 33H, Function 18H	Mouse
Install alternate event handler	

This function permits a program to install a limited range event handler. This handler can be called by the mouse driver when certain mouse events occur in conjunction with the keyboard.

Input

AX = 0018H

CX = Events which should trigger the call of the event handler

Bit 0: Mouse movement

Bit 1: Left mouse button activated

Bit 2: Left mouse button released

Bit 3: Right mouse button activated

Bit 4: Right mouse button released

Bit 5: Shift key pressed during mouse button event

Bit 6: Ctrl key pressed during mouse button event

Bit 7: Alt key pressed during mouse button event

Bits 8-15: Unused

ES = Segment address of event handler

DX = Offset address of event handler

Output

AX = Installation status

AX=0018H: Event handler installed

AX=FFFFH: Event handler could not be installed

Remarks

At least one of bits 5 to 7 must be set in the event mask of the CX register to ensure that the event reacts to at least one of the control keys. If the programmer prefers not to read the Shift, Ctrl or Alt keys along with mouse buttons, use functions 0CH or 14H instead.

An error can occur if three alternate event handlers were previously installed, or if an event handler with the same event mask already exists.

Remarks

The event handler is called by the mouse driver through a FAR call assembler instruction, and therefore must be terminated with a FAR RET instruction. None of the various processor registers may be returned to the caller with a changed content.

The mouse driver passes the following information to the event handler through the processor registers during the call:

AX = Event mask. The bits correspond to the various events as indicated in the CX register during the installation of the event handler. In addition, other bits can be set, since the value reflects the current status of the mouse driver, and is not limited to the selected events.

BX = Mouse button status:

Bit 0 = Left mouse button activated

Bit 1 = Right mouse button activated

Bit 2 = Center mouse button activated

CX = Horizontal mouse position.
 DX = Vertical mouse position.
 SI = Length of last horizontal mouse movement.
 DI = Length of the last vertical mouse movement.
 DS = Data segment of the mouse driver.

The coordinates returned in the CX and DX registers refer to the pixel positions in the virtual mouse display screen rather than physical positions on the actual display screen.

The values in the SI and DI registers refer to mickeys (one mickey = 1/200 inch).

These mickey values must be interpreted as signed numbers. Positive values indicate movement toward the bottom or right border of the screen, while negative values indicate movement toward the top or left border of the screen.

Interrupt 33H, Function 19H	Mouse
Determine address of alternate event handler	

Returns the address of an alternate event handler to the caller.

Input	AX = 19H
	CX = Event handler event mask
Output	CX = 00H: Error
	ES = Segment address of event handler
	DX = Offset address of event handler

Remarks

See the description of function 18H above for additional information about the meanings of each bit in the event mask.

The function call fails if no alternate event handler with the indicated event mask was previously installed.

Interrupt 33H, Function 1AH	Mouse
Set mouse sensitivity	

Defines the relationship between physical mouse movement and mouse cursor movement. Also defines the maximum for doubling mouse speed.

Input	AX = 1AH
	BX = Number of horizontal mickeys
	CX = Number of vertical mickeys
	DX = Maximum limit for doubling the mouse speed
Output	No output

Remarks

Values in the CX and DX registers can range from 1 to 32767.

The default setting is 8 horizontal mickeys and 16 vertical mickeys. This causes the mouse cursor to move twice as fast horizontally as it moves vertically.

To prevent doubling of the mouse speed, the limit can be set higher.

Speeds in excess of 5,000 mickeys per second cannot be achieved by practical means.

Calling function 00H (Reset mouse driver) changes any previously set values to the default values.

Interrupt 33H, Function 1BH	Mouse
Determine mouse sensitivity	

Returns the parameters previously set by calling function 1AH or functions 0FH and 13H.

Input	AX = 1BH
Output	BX = Number of horizontal mickeys
	CX = Number of vertical mickeys
	DX = Maximum limit for doubling the mouse speed

Interrupt 33H, Function 1CH	Mouse
Set mouse hardware interrupt rate	

Determines the frequency at which the mouse hardware reads the current mouse position and mouse button status.

Input	AX = 1CH
	BX = Interrupt rate
	Bit 0: No interrupts
	Bit 1: 30 interrupts per second
	Bit 2: 50 interrupts per second
	Bit 3: 100 interrupts per second
	Bit 4: 200 interrupts per second
	Bits 5-15: Unused
Output	No output

Remarks

This function is only available for the Inport mouse.

If more than one bit is set in the BX register, only the least significant bit which is set counts.

The mouse's resolution increases with the number of interrupts. The increased number of mouse interrupts decreases the speed of the foreground program.

Interrupt 33H, Function 1DH	Mouse
Set display page	

Specifies the display page on which the mouse cursor appears.

Input AX = 1DH
 BX = Number of the display page

Output No output

Remarks

Default value is display page 0.

Calling this function only makes sense if the application program works with several display pages, as available on CGA, EGA and VGA cards.

Interrupt 33H, Function 1EH	Mouse
Determine display page	

Determines the display page on which the mouse cursor appears.

Input AX = 1EH

Output BX = Number of the display page

Interrupt 33H, Function 1FH	Mouse
Disable mouse driver	

Deactivates the current mouse driver and returns the address of the previous interrupt handlers for interrupt 33H.

Input AX = 1FH

Output AX = Error status
 AX=FFFFH: Error
 AX=1FH: O.K.
 ES = Segment address of previous event handler
 BX = Offset address of previous event handler

Remarks

This call releases any previously installed and active mouse driver interrupt routines. The exception to this is the handler for interrupt 33H, but the caller can reload this interrupt vector with its original value since this address is returned in the ES:BX register pair.

Interrupt 33H, Function 20H	Mouse
Enable mouse driver	

Activates a mouse driver previously deactivated by function 1FH.

Input AX = 20H

Output No output

Interrupt 33H, Function 21H	Mouse
Reset mouse driver	

Resets the mouse driver, disables the mouse cursor and disables the currently installed event handler.

Input AX = 21H

Output AX = Error status

AX=FFFFH: Error

AX=0021H: O.K.

BX = Number of mouse buttons

Remarks

Unlike function 00H, this function does not perform a total mouse hardware reset.

Interrupt 33H, Function 22H	Mouse
Set language for messages	

Set language for messages

Specifies language for mouse messages.

Input AX = 22H

BX = Language number

BX=0: English

BX=1: French

BX=2: Dutch

BX=3: German

BX=4: Swedish

BX=5: Finnish

BX=6: Spanish

BX=7: Portuguese

BX=8: Italian

Output No output

Remarks

This function applies only to the mouse driver published for international use. Function 22H is not available on the domestic version of the mouse driver.

Interrupt 33H, Function 23H	Mouse
Get language number	

Returns the number indicating the language under which the mouse driver is operating.

Input AX = 23H

Output BX = Language number

BX=0: English

BX=1: French

BX=2: Dutch

BX=3: German

BX=4: Swedish

BX=5: Finnish

BX=6: Spanish

BX=7: Portuguese

BX=8: Italian

Remarks

This function applies only to the mouse driver published for international use. Function 23H is not available on the domestic version of the mouse driver.

Interrupt 33H, Function 24H	Mouse
Determine mouse type	

Determines the type of mouse installed and the version number of the mouse driver.

Input AX = 24H

Output BH = Whole number of the version number

BL = Fraction of the version number

CH = Mouse type

CH=1: Bus mouse

CH=2: Serial mouse

CH=3: Inport mouse

CH=4: PS/2 mouse

CH=5: HP mouse

CL = IRQ number

CL=0: PS/2

CL=2, 3, 4, 5 or 7: IRQ number in the PC

Remarks

If the version number of the mouse driver is for example 6.24, the value 6 is returned in the BH register and the value 24 is returned in the BL register.

Interrupt 33H, Function 25H

Mouse (Version 6.26 and above)

Get general driver information

Returns general information describing the mouse driver, such as the driver type and cursor type.

Input

AX = 25H

Output

AX = General information (see below)

BX = OS/2 status information

CX = OS/2 status information

DX = OS/2 status information

Remarks

The AX register receives the general information as a bit field. These bits have the following meanings:

Bit 15: Type of driver

0(b)=COM file

1(b)=Device driver accessed through CONFIG.SYS

Bit 12 and 13: Mouse cursor information

00(b)=Software text cursor

01(b)=Hardware text cursor

10(b), 11(b)=Graphic cursor

Bits 8 to 11 : Mouse hardware interrupt rate

The arguments returned in the BX, CX and DX registers apply only to mouse drivers in OS/2. They have no significance in DOS programming.

Interrupt 33H, Function 26H	Mouse (Version 6.26 and above)
Get maximum virtual coordinates	

Returns the maximum virtual mouse display coordinates, and indicates whether the mouse driver is active or inactive.

Input	AX = 26H
Output	BX = Mouse driver status BX=0: Inactive BX>0: Active CX = Maximum virtual X-coordinate DX = Maximum virtual Y-coordinate

Remarks

Functions 1FH and 20H control the mouse driver status, returned in the BX register after calling this function.

The values returned in the CX and DX registers describe the size of the virtual mouse display screen, not the cursor positions specified in functions 07H and 08H.

Interrupt 33H, Function 27H	Mouse (Version 7.01 and above)
Get masks and mickey counts	

Reads screen and cursor masks. Also, this function returns information about mouse movement since the last reading.

Input	AX = 27H
Output	AX = AND mask (software cursor) or starting scan line (hardware cursor) BX = XOR mask (software cursor) or ending scan line (hardware cursor) CX = Length of horizontal movement in mickeys DX = Length of vertical movement in mickeys

Remarks

The values returned vary with the type of cursor active during the function call. If the hardware cursor is active, the function receives the starting and ending scan lines of the cursor. If a software cursor is active, the function receives the current screen and cursor mask values.

This function has been supported since Version 7.01 of the mouse driver. Version 7.02 was the first version to return the scan line information describing the hardware cursor.

The movement values returned in the CX and DX registers are taken directly from the mouse hardware, and are not influenced by various software settings such as the threshold value for doubling mouse speed, or the acceleration curve.

Interrupt 33H, Function 28H	Mouse (Version 7.0 and above)
Set video mode	

Sets the video mode if the selected mode is supported by the active video card.

Input

AX = 28H

CX = Video mode number

DX = Screen font size

Output

CX = Function status

Remarks

A list of the available video modes and their code numbers can be queried with function 29H.

The value zero is returned in the CX register if the video mode indicated is supported by the active video-hardware and therefore could be set. Otherwise the code number from the AX register is returned.

The calling parameter in DX is only expected with a few video modes which operate with settable font sizes. In the higher level byte of DX the size of the font, together with the Y-axis and in the lower level byte the extent along the X-axis, must be coded.

Interrupt 33H, Function 29H**Mouse** (Version 7.0 and above)

Count video modes

Gets a numbered list of video modes supported by the active video card.

Input

AX = 29H

CX = Video mode

CX=0: First video mode

CX<>0: Next video mode

Output

BX = Segment of string

CX = Video mode number

DX = Offset of string

Remarks

Multiple calls of this function are required to generate a complete list of supported video modes. The value 0 must be passed in the CX parameter for the first call, with values unequal to 0 passed for any subsequent calls.

Each subsequent call describes a video mode. When the CX register returns 0, this indicates that all available video modes have been read.

Function 29H doesn't directly return the video mode's type (text or graphic), resolution or color capability. This information can usually be obtained from an ASCII string whose address is returned in the BX:DX register pair. If this ASCII string is available, and if the BX:DX registers contain values other than 0, the ASCII string ends with a dollar sign and a null byte.

Interrupt 33H, Function 2AH**Mouse** (Version 7.02 and above)

Get cursor hotspot

Returns information about the mouse type and cursor hotspot.

Input	AX = 2AH
Output	AX = Internal cursor flag
	BX = Hotspot X-coordinate
	CX = Hotspot Y-coordinate
	DX = Type of mouse

Remarks

The internal cursor flag indicates whether the mouse cursor is visible or not. Functions 01H and 02H indirectly influence this flag. A value of 0 signals that the mouse cursor is currently invisible. Any other value indicates that the mouse cursor is currently visible.

The hotspot is the pixel in a graphic cursor mask whose position is returned when reading cursor position. Its distance from the upper-left corner of the bit mask is returned to the BX and CX registers as a signed integer. This integer can range from -128 to 127).

The DX register indicates the mouse type. This type code can be one of the following:

- 0 = No mouse
- 1 = Bus mouse
- 2 = Serial mouse
- 3 = InPort mouse
- 4 = PS/2 mouse
- 5 = Hewlett-Packard mouse

Interrupt 33H, Function 2BH**Mouse** (*Version 7.0 and above*)

Set acceleration curves

All four acceleration curves, which the mouse-driver administers internally, can be loaded with the help of this function and one can be selected as the current one.

Input	AX = 2BH
	BX = Number of curve to activate
	ES = Segment address of curve array
	SI = Offset address of curve array

Output AX = FFFFH: Error

AX = 0000H: O.K.

Remarks

This function changes the preset acceleration curves in the driver, by passing the value -1 as the number of the current acceleration curve. The passing of a buffer with the data of the acceleration curves is not required in this case.

The four acceleration curves are described through a data structure which is created by the caller in memory and must be passed using the ES:SI registers. See Chapter XXX for more information about this structure.

Interrupt 33H, Function 2CH	Mouse (Version 7.0 and above)
Read acceleration curves	

Reads the acceleration curves.

Input	AX = 2CH
Output	AX = 00H: O.K. BX=Number of the current acceleration curve (0 to 3) ES=Segment address of curve array SI=Offset address of curve array AX = FFFH: Error

Remarks

The AX register should be read following every call of this function. The acceleration curves could only be read if the AX register contained a value of 0.

The contents of the ES:SI register pair indicate the buffer containing the array describing the current acceleration curve. This data structure corresponds to the format used in function 2BH for setting an acceleration curve.

Interrupt 33H, Function 2DH	Mouse (Version 7.0 and above)
Get/set active acceleration curves	

Activates one of the four acceleration curves, and reads the current acceleration curve.

Input	AX = 2DH BX = -1: Get current acceleration curve BX = 1 - 4: Set current acceleration curve
Output	AX = 0: O.K. BX=Number of current active acceleration curve ES=Segment address of ASCII string describing acceleration curve SI=Offset address of ASCII string describing acceleration curve AX=-2: Bad curve number

Remarks

If the value -1 is passed in the BX register as part of the function call, this function returns information describing the current acceleration curve in the BX, ES and SI registers. If a value from 1 to 4 is passed in the BX register, the corresponding acceleration curve becomes active. If this is the case, the BX register returns the current acceleration curve number.

After setting the current acceleration curve, the ES:SI registers indicate the ASCII string containing acceleration curve data (see function 2BH). This string contains 16 bytes, and has no special end character (i.e., null byte or \$). This string provides the symbolic name of the acceleration curve.

Interrupt 33H, Function 2EH	Mouse (<i>Version 1.0 and above</i>)
Undocumented	

This undocumented function is used exclusively by the mouse driver, and cannot be called from a program.

Interrupt 33H, Function 2FH	Mouse (<i>Version 7.02 and above</i>)
Mouse hardware reset	

Resets the mouse hardware without affecting the software configuration (mouse cursor appearance, threshold value, acceleration curve settings, etc.).

Input AX = 2FH

Output AX = FFFFH: O.K.

 AX = 0: Error

Remarks

This function is the hardware equivalent of function 21H, which resets the software parameters specified in the mouse driver.

Interrupt 33H, Function 30H	Mouse (<i>Version 7.04 and above</i>)
Get/set ballpoint information	

This function has been tailored specifically for the needs of the ballpoint mouse.

Input AX = 30H

 BX = Angle of rotation

 CX = Command code

Output AX = Function status

 BX = Angle of rotation

 CX = Active buttons

Remarks

After the function call, the function status in AX should be checked immediately, because the value -1 indicates that no Ballpoint-Mouse is installed. Any other value indicates however the existence of a Ballpoint-Mouse and reflects the status of the various mouse-buttons. The individual buttons are represented by the following bits in the AX register:

Bit 2 = Button 4

Bit 3 = Button 2

Bit 4 = Button 3

Bit 5 = Button 1

The remaining bits contain the value 0.

If the current angle of rotation and the active buttons are queried with this function, the CX register must be loaded with the value zero before the function call. An angle of rotation in BX is not required at this time.

As a function result, the angle of rotation is returned in the BX register. It is a value between 0 and 360 (degrees). The two active buttons can be read from the high byte of CX, while the inactive buttons are coded into the low byte of this register. In these two bytes are the following bits for the individual buttons:

Bit 2 = Button 4

Bit 3 = Button 2

Bit 4 = Button 3

Bit 5 = Button 1

the remaining bits contain the value 0.

If the mouse-driver should be informed with the help of this function of the current angle of rotation of the Ballpoint-Mouse and the two active buttons selected, the active and inactive buttons must be coded into the CX register. The coding is exactly as in the return for the active and inactive buttons after a query (see above). In addition, a value between 0 and 360 degrees is expected as the angle of rotation in BX.

Interrupt 33H, Function 31H	Mouse (Version 7.05 and above)
Get minimum/maximum virtual coordinates	

Returns current minimum and maximum coordinates of the virtual mouse display screen in the current video mode.

Input	AX = 31H
Output	AX = Minimum X-coordinate BX = Minimum Y-coordinate CX = Maximum X-coordinate DX = Maximum Y-coordinate

Remarks

Functions 07H and 08H affect the size of the virtual mouse display screen.

Interrupt 33H, Function 32H	Mouse (Version 7.05 and above)
Get active advanced functions	

Returns information describing advanced functions supported by the mouse driver, and not accessible from function 25H.

Input	AX = 32H
Output	AX = Supported functions

Remarks

The function result in AX is a bit field, where each bit stands for a function. If it is set, the function is supported.

Remarks

The AX register receives the function support as a bit field. These bits have the following meanings:

Bit 15=Function 25H

Bit 14=Function 26H

Bit 13=Function 27H

Bit 12=Function 28H

Bit 11=Function 29H

Bit 10=Function 2AH

Bit 9=Function 2BH

Bit 8=Function 2CH

Bit 7=Function 2DH

Bit 6=Function 2EH

Bit 5=Function 2FH

Bit 4=Function 30H

Bit 3=Function 31H

Bit 1=Function 32H

Bit 0=Function 33H

Interrupt 33H, Function 33H**Mouse** (*Version 7.05 and above*)

Get switch settings

Returns all mouse parameters in the mouse driver that can be set through hardware or software.

Input

AX = 33H

CX = Buffer length

ES = Segment address of buffer

DX = Offset address of buffer

Output

AX = 0

CX = Number of bytes in buffer

ES = Segment address of buffer

DX = Offset address of buffer

Remarks

The buffer has the following structure:

Offset	Content	Range
0	Mouse type (low nibble)	0-5
0	Mouse type (high nibble)	0-4
1	Language	0-10
2	Horizontal sensitivity	0-100
3	Vertical sensitivity	0-100
4	Double threshold	0-100
5	Ballistic curve	1-4
6	Interrupt rate	1-4
7	Cursor override mask	0-255
8	Laptop adjustment	0-255
9	Memory type	0-2
10	Super VGA support	0-1
11	Rotation angle	0-359
13	Primary button	1-4
14	Secondary button	1-4
15	Click lock enabled	0-1
16	Acceleration curve data	Bytes 16-339

Interrupt 33H, Function 34H**Mouse** (*Version 8.0 and above*)

Get MOUSE.INI location

Returns the exact path designation of the MOUSE.INI file as an ASCII string. This function applies only to Microsoft Windows.

Input AX = 34H

Output AX = 0

ES = Segment address of buffer

DX = Offset address of buffer

Remarks

The ASCII string is terminated by a null byte.

The path of MOUSE.INI is obtained from the MOUSE variable. If this variable was not defined, MOUSE.INI is assumed to be in the directory containing a mouse driver.

Hardware Interrupts

Interrupt 00H	Hardware (CPU)
Division by zero	

The CPU calls this interrupt when it encounters a divisor of 0 during one of the two assembly language division instructions (DIV or IDIV). According to the rules of mathematics, dividing a number by 0 is illegal. During the booting process, this interrupt points to a routine that, when called, displays the "Division by Zero" error message (or a similar message) on the screen. The interrupt continues with the execution of the current program.

Interrupt 01H	Hardware (CPU)
Single step	

The CPU calls this interrupt when the TRAP bit in the flag register of the CPU has been set to 1. Then the interrupt is called after the execution of each assembly language instruction. This allows the user to follow these instructions, determine the changes in register contents and determine which instructions are executed. To prevent the call of the interrupt after the execution of every instruction in the trap routine (which would create an endless loop and a stack overflow), the processor resets the TRAP bit upon entry to the trap routine. If the trap routine ends with the IRET instruction, it automatically resets the TRAP bit to its old value by restoring the complete flag register from the stack. Because of this, the execution of the next instruction calls interrupt 1 again. Once the programmer has obtained the necessary information about a program from single step mode, the TRAP mode (or TRAP bit) can be disabled.

Interrupt 02H	Hardware (CPU)
NMI	

The hardware calls this interrupt when an error is discovered in the RAM chips. The system calls the non-maskable interrupt because this type of error impairs the capabilities of the system, and can lead to a crash. The NMI has the highest priority of all interrupts and therefore is executed faster than other interrupts. The NMI usually calls a BIOS routine which informs the user of a memory error, lists the number of defective memory chips and stops the system.

If the NMI detects an error, the math coprocessor included in some PCs can also trigger the NMI. Even though NMI usually cannot be suppressed, the PC allows an exception to this rule. Some PC/XT and AT models have a special port (port A0H on PCs and XTs, port 70H on ATs). If a 0 value is written to one of these ports, the NMI interrupt is disabled. If the ports return the value 80H, the NMI interrupt is enabled.

Interrupt 03H	Hardware (CPU)
Breakpoint	

While the other interrupts can be called with a two-byte assembly language instruction (first byte CDH, second byte the number of the interrupt), interrupt 3 is called by the single-byte instruction CCH. This interrupt can be used to test programs when you want to execute the program up to a certain instruction, then stop and display the current register contents. Utilities

designed for program testing like DEBUG implement this by placing calls for interrupt 3 where the break should occur. When the program is executed and the processor reaches the instruction, it calls interrupt 3. The program testing utility contains a routine which displays the register contents and other information.

Interrupt 04H	Hardware (CPU)
Overflow	

This interrupt can be called by the INTO (INTerrupt on Overflow) conditional assembly language instruction. The call occurs when the overflow bit in the flag register is set during the execution of the INTO instruction. This can happen following math operations (e.g., multiplication with the MUL instruction) that produce a result which cannot be represented within a specified number of bits. This interrupt can also be called with the normal INT instruction, but this instruction isn't controlled by the status of the set overflow bit. Since it is seldom used, DOS points this interrupt to an IRET instruction.

Interrupt 05H	BIOS
Hardcopy	

BIOS calls this interrupt when the user presses the **Prt Sc** key. The system then makes a hardcopy by sending the current screen contents to a printer. BIOS initializes the interrupt vector from the vector table and points to the BIOS hardcopy routine in ROM-BIOS. Assembly language and programs written in higher .High level languages can use this interrupt with the INT instruction to get a hardcopy during program execution.

Interrupt 08H	Hardware (8259 interrupt controller)
Timer	

In the PC, the 8259 timer chip receives 1,193,180 signals per second from the heart of the system, which is an oscillating quartz crystal. After 65,536 of these signals (1 second), it triggers a call of interrupt 8, which the 8259 transmits to the CPU. Since the frequency of the call of this interrupt is independent of the system clock frequency, interrupt 8 works well for timekeeping. The PC also uses the interrupt for timekeeping. BIOS points the interrupt vector of this interrupt to its own routine, which is called 18.2 times per second. A time counter increments every second and disables the disk drive motor if disk access hasn't occurred within a certain time period.

Interrupt 09H	Hardware (8259 interrupt controller)
Keyboard	

PC keyboards contain an independent processor. This Intel processor carries either the number 8048 (PC/XT) or 8042 (AT). This processor monitors the keyboard and registers whether a key was depressed or released. When either of these actions occur, this processor must inform the CPU so the code of the activated key can be sent to the system and processed. The keyboard instructs the interrupt controller to call interrupt 9. This interrupt calls a BIOS routine that reads the character from the keyboard and places it into the keyboard buffer.

Introduction To Number Systems

We've often mentioned numbers in the binary system and hexadecimal systems instead of the normal decimal system. This Appendix presents a brief introduction to these number systems.

Decimal system

Before explaining the new number systems, you should know the basic concepts of the decimal system. The decimal number 1989 can also be written as $1*1000+9*100+8*10+9*1$. This shows that if you number the digits from right to left, the first number represents a column of ones, the second number represents a column of tens, the third number represents a column of hundreds and the fourth number represents a column of thousands. The numbers increase from right to left in powers of 10.

The first digit of any number system has the value 1. The factor by which the value increases from one column to the next differs among the number systems. This factor corresponds to the numbers with which the number system works. The factor is 10 with the decimal system because ten different numbers are available for each digit (0 to 9).

This principle of powers for each column also applies to the binary and hexadecimal systems.

Binary system

Since a computer recognizes the numbers 0 and 1 on its lowest functional level, the binary system is essential to computing. The value of the numbers double from column to column because the binary system only uses powers of two for each column (i.e., the numbers 0 and 1 instead of the numbers 0 to 9).

Now let's count the binary places starting from right to left as we did in the decimal example described above. The first (right hand) position counts as one, the second as two, the third as four and the fourth as eight. The places then follow as 16, 32, 64, 128, etc.

For example, 11001 binary converts to 25 decimal, or the equation $1*16+1*8+0*4+0*2+1*1$.

Hexadecimal system

Unlike the binary system, the hexadecimal system; operates with more basic numbers than the decimal system. This system counts single digits from 0 to F. Since only the ten numbers of the decimal system are able to represent a number, the numbers from 10 to 15 in hexadecimal use the letters A to F in addition to the numbers 0 to 9. AH represents 10, BH for 11, CH for 12, DH for 13, EH for 14 and FH for 15.

By using 16 numbers or letters for each position, the value by which each position increments is 16.

The first position has the value 1, the second 16, the third 256 and the fourth 4,096.

For example, the hexadecimal number FB3H converts into 4,019 decimal, or $15*256+11*16+3*1$.

Hex and binary

The hexadecimal system and the binary system are easily converted back and forth. For example, one four-digit binary number converts to a single-digit hexadecimal number. Because of this, the hexadecimal system is an important part of assembly language programming. It's much simpler to convey a byte (an eight-bit number) using two hexadecimal digits than it is for the developer to compute a 16-bit binary equivalent.

This book denotes all binary numbers by the letter b, and all hexadecimal numbers by the letter H.

The following tables should help explain number systems more clearly.

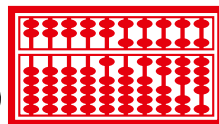
Number of positions in each number system					
	P				
Decimal	10000	1000	100		
Binary	16	8	4	2	1
Hexadecimal	65536	4096	256	16	1

Comparing selected numbers in each number system		
Dec		
0	0(b)	0H
1	1(b)	1H
2	10(b)	2H
3	11(b)	3H
4	100(b)	4H
5	101(b)	5H
6	110(b)	6H
7	111(b)	7H
8	1000(b)	8H
9	1001(b)	9H
10	1010(b)	AH
11	1011(b)	BH
12	1100(b)	CH
128	10000000(b)	80H
129	10000001(b)	81H
256	100000000(b)	100H
1024	10000000000(b)	400H
4096	1000000000000(b)	1000H
65535	1111111111111111(b)	FFFFH

Computer Books & Software

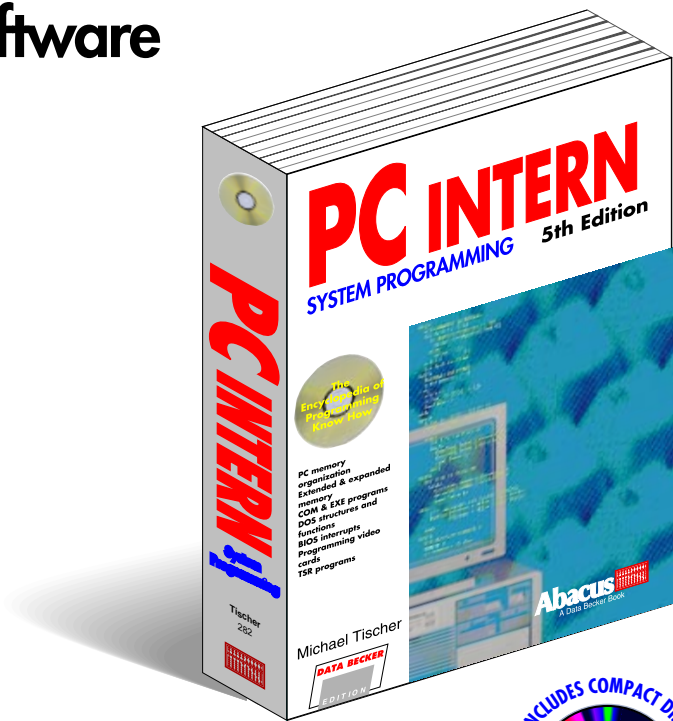
Catalog

Abacus

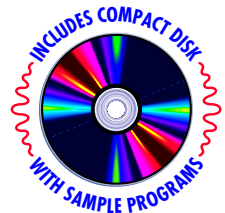


PC catalog

Order Toll Free 1-800-451-4319
Books and Software



Abacus 



 **To order direct call Toll Free 1-800-451-4319**

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Dear Computer Enthusiast,

For over 15 years, millions of readers like yourself have relied on Abacus for quality computer info. Recall that we're one of the pioneers of book/ disk and book/ CD-ROM combinations. With these combos, users can read about then use the accompanying software immediately.

This latest catalog has many new titles such as "The CD-ROM Book" and "The PC Video Book" to stay abreast of the fast-changing technologies. For more relaxing times, "The Stereogram Book and Software" is sure to treat your eyes to some fascinating 3D surprises.


For the more technical reader, we're proud to announce that our critically acclaimed "PC Intern" is now in its 5th Edition. With more than 500,000 copies of this book sold worldwide, it's the most authoritative reference for every PC programmer. This new edition also includes a companion CD-ROM.

Whatever your interests, we hope you'll find a title that will fill your needs. Thanks for your continued patronage. And remember - "you can count on Abacus".

Happy computing,

Arnie Lee
President






Includes
ready-to-use
companion diskette

Save Time & Effort

When you see either of these symbols next to a book it means that a companion disk or CD-ROM is included with the book. We helped to pioneer book/ disk combinations. Readers tell us they save time & effort by utilizing the software immediately, without having to type in programs or text.



INCLUDES COMPACT DISK



3 Easy Ways to Order



CALL our sales department at **1-800-451-4319** in US & Canada, Monday through Friday 9:00 am to 5:30 pm (Eastern Time).



FAX your completed order form to: **(616) 698-0325**, 24 hours a day



MAIL the order form along with check, money order, or credit card information payable in US dollars to:

Abacus • 5370 52nd Street SE • Grand Rapids, MI 49512 USA

Developers Series books are for professional software developers who require in-depth technical information and programming techniques.

PC Intern *System Programming* **5th Edition**

PC Intern is a completely revised edition of our bestselling *PC System Programming* book for which sales exceeded 100,000 copies. **PC Intern** is a literal encyclopedia for the PC programmer. Whether you program in assembly language, C, Pascal or BASIC, you'll find dozens of practical, parallel working examples in each of these languages. **PC Intern** clearly describes the technical aspects of programming under DOS.

Some of the topics covered include:

- PC memory organization
- Using extended and expanded memory
- Hardware and software interrupts
- Programming for networks
- Handling program interrupts in BASIC, Turbo Pascal, C and Assembly Language
- DOS extensions and DPMI/VCPI
- Using BIOS and interrupts for I/O operations
- Graphic programming
- TSR programs
- Interfacing with DOS 6 and DoubleSpace
- CD-ROM technology

The companion CD-ROM contains complete text indexed and all program code in the book. The CD-ROM lets you find information even faster, making **PC Intern** a state- of- the- art digital reference.

Author: Michael Tischer

Order Item: #B282

ISBN: 1-55755-282-7

Suggested retail price: \$59.95 US/ \$75.95 with CD-ROM



To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

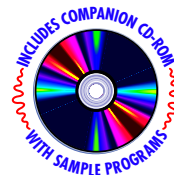
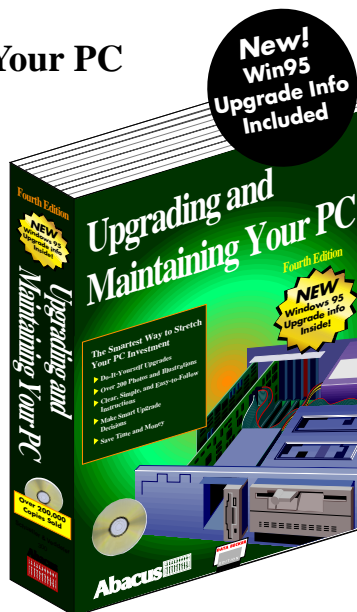
**Productivity Series books are for users who want
to become more productive with their PC.**

Upgrading and Maintaining Your PC **New 4th Edition**

Buying a personal computer is a major investment. Today's ever-changing technology requires that you continue to upgrade if you want to maintain a state of the art system. Innovative developments in hardware and software drive your need for more speed, better hardware, and larger capacities. **Upgrading and Maintaining Your PC** gives you the knowledge and skills that help you upgrade or maintain your system. You'll save time and money by being able to perform your own maintenance and repairs.

How to upgrade, repair, maintain and more:

- Hard drives, Memory and Battery Replacement
- Sound & Video Cards, CD-ROM's
- Pentium Powerhouses 60 - 133 MHz, overdrive processor
- Large Capacity Hard Drives
- Quad Speed CD-ROM Drives
- Sound Cards - 64-bit and Wave Table
- Modems/Fax Cards, ISDN
- AMD, Cyrix, AMI and Intel Processors
- Operating Systems - DOS 6.22, Novell DOS, IBM PC DOS 6.3, OS/2 Warp & Windows 3.1 to Windows 95



On the CD-ROM!

System Sleuth Analyzer from Dariana (\$99.95 value) - A toolbox of valuable PC diagnostic aids rolled into a single, easy-to-use software utility. It lets you explore exacting details of your PC without fear of accidental, unrecoverable modifications to a particular subsystem.

Cyrix Upgrade Compatibility Test - Run "Cyrix's" own test to see if you can upgrade with one of their new 486 chips!

Intel's Pentium Chip Test - calculate the now famous "math problem" on your own system!

Authors: H. Veddeler & U. Schueller

Order Item:#B300

ISBN: 1-55755-300-9

Suggested retail price: \$34.95 US / \$46.95 CAN with CD-ROM

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Productivity Series books are for users who want to become more productive with their PC.

Win95 Rx

Solve Your Own Windows 95

Computing Problems Easily

Solve all of your computing problems once and for all. Here's an easy-to-follow troubleshooting guide that will save you time, money and headaches. Learn how to identify and fix problems yourself instead of having to endlessly wait for vague, unclear answers from those so-called "expert" tech support lines. Do it right. Do it yourself!

What's inside the book:

- Preparing your system for Windows 95
- Installing new or additional hardware
- Tips for working with the new desktop
- Protecting your valuable data – backup and viruses
- Tune up your hard drive for peak performance
- Multimedia setup and software
- Running your existing DOS and Win 3.X programs
- Running your DOS games without the nightmares

Win95 Rx includes a companion CD-ROM filled with useful tools for testing and configuring your entire PC system.

Author: Kober, Buechel & Baecker

Order Item: #B297

ISBN: 1-55755-297-5

Suggested retail price: \$34.95 US/ \$46.95 with CD-ROM



To order direct call Toll Free 1-800-451-4319

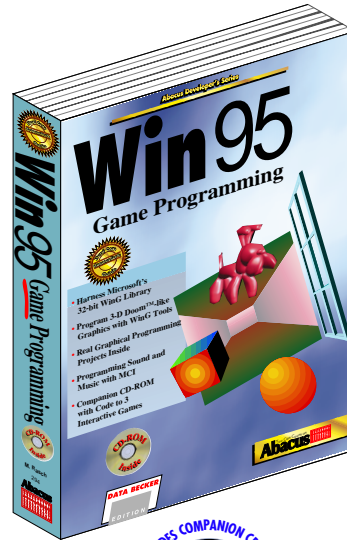
In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Developers Series books are for professional software developers who require in-depth technical information and programming techniques.

Win95 Game Programming Hard Core Gamewriter's Guide

Windows 95 is going to revolutionize the way programmers code graphics, sound and music. The new 32-bit WinG Library from Microsoft is a giant toolbox of graphics capabilities that insulate the gamewriter from the hardware, yet delivers the performance that you need to write fast, responsive games and animations.

Win95 Game Programming introduces you to WinG with practical, working examples. You'll learn about "Windowing" with APIs, creating graphic effects with multichannel sound (MIDI and Wave files), applying texture mapping, light and shadow effects and more! The book takes you through three complete projects to demonstrate the graphical programming techniques.



Game Programming Projects in this book:

- Jump - A Classic in New Clothing
- Space Flight Simulator - From the Virtual World to the Screen
- Underworld - A DOOM™-like game
- Underground Designs & Descent into the Caves

You'll find all the source code to these exciting game programs on the CD! Written in Pascal and C, you can use this code to work on your own graphic projects and games.

Author: Matthias Rasch

Order Item: #B294

ISBN: 1-55755-294-0

Suggested retail price: \$44.95 US/ \$59.95 with CD-ROM

To order direct call Toll Free 1-800-451-4319

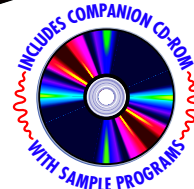
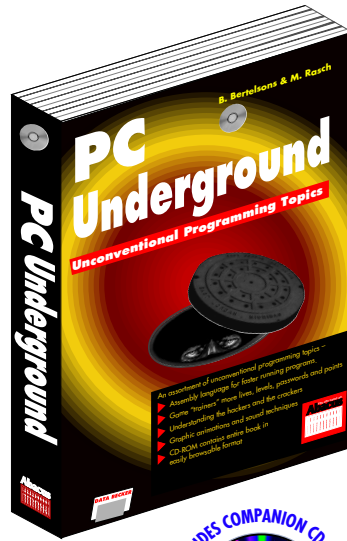
In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

**Productivity Series books are for users who want
to become more productive with their PC.**

PC Underground

Unconventional Programming Techniques

PC Underground is not a criminal's handbook; instead it's an in depth programmer's guide to system level programming. It covers a diverse range of unconventional programming topics related to: memory management techniques, direct parallel port programming, sound card programming (Sound Blaster and Gravis Ultrasound cards), assembly language techniques, and more. It includes extensive coverage of graphics - programming the VGA card, creating 3D representations and effects, working the image formats and the high resolution modes.



This book also shows you how crackers and hackers cruise the PC world looking for openings. To understand the "cracker's" mind, you have to see how he operates. After introducing you to the world of crackers, it then shows you hard core programming techniques. You'll see how "game trainers" are programmed to manipulate numbers of lives, scores and levels for example. You'll learn insider details about password protection and other security mechanisms.

The CD-ROM with this book contains numerous graphics and sound demos, actual program code described in the book, a game on which you can try a "trainer", a series of helpful tools and several shareware utilities.

Author: B. Bertelons & M. Rasch

Order Item: #B275

ISBN: 1-55755-275-4

Suggested retail price: \$34.95 US / \$44.95 CAN includes CD-ROM

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Multimedia Presentation

The Photo CD Book

The Photo CD Book is for all CD-ROM owners, multimedia enthusiasts, graphic artists, desktop publishers and users wanting to grasp the movement to Photo CDs and this exciting new digital image technology.

This book is your first source for learning and understanding the digital technology invented by Eastman Kodak to "replace" conventional photography. It's a "hands-on" way to learn and use the technology where you'll see how to turn your personal computer into a digital photo studio, darkroom lab and art gallery. From the basics of image processing and the tricks of photo retouching and enhancement, to the radical special effects of morphing and then finally to the on-screen exhibition halls for visual presentations, you'll get a complete coverage of where this technology is taking us.

Learn how the different hardware is used to record, manipulate and output Photo CD images. You'll also see how the various software packages can transform ordinary into sensational images. One section describes how to use Photo CD images with Micrografx PhotoMagic, Microsoft Publisher, CorelDraw, Aldus FreeHand and Photostyler, Adobe PhotoShop, and other popular software.

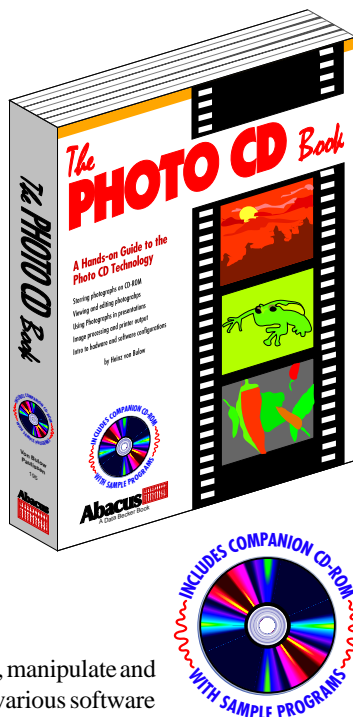
The companion CD-ROM has dozens of samplers including Toshiba's Photo/Graphic Viewer; MicroTek's Photo Star graphic utility; real PCD photo examples used in the book; a collection of the most popular shareware graphics utilities including PaintShop Pro and Graphics WorkShop; several industry standard Phillips CD-I software drivers.

Author: Heinz von Buelow and Dirk Paulissen

Order Item: #B195

ISBN: 1-55755-195-2

Suggested retail price: \$29.95 US / \$39.95 CAN includes CD-ROM



To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Multimedia Presentation

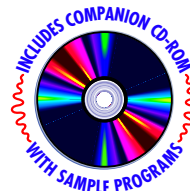
The PC Video Book

The PC Video Book teaches you the fundamentals of video technology and the hardware for creating your own videos on the PC. Quickly learn how to put credits in videos, how to adjust video color and write it back to the tape, and much more. It's the complete guide to video production including- video planning, digitizing, editing, changing color, adding effects, titles, and credits, etc. From the fundamentals of video technology to creating your own video productions- this book offers valuable tips, concrete shopping suggestions, and helpful explanations.



The book includes:

- Video production step-by-step
- Frame grabbing for desktop video
- Using Video for Windows
- File formats: AVI, FLC, FLI, DIB, WAV, PCM
- Creating and recording animations
- Capturing video and sound
- Choosing and connecting VCRs, camcorders and other hardware
- Editing video and sound clips



On the CD-ROM are hundreds of megabytes of digitized material (videos, images, sounds), which can be tried out and used in multimedia applications.

Authors: Kerstin Eisenkolb & Helge Weickardt

Order Item: #B265

ISBN: 1-55755-265-7

Suggested retail price: \$34.95 US / \$44.95 CAN includes CD-ROM

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

**Productivity Series books are for users who want
to become more productive with their PC.**

Electronic Alphabet- **Using TrueType Fonts for More Creative Desktop Publishing**

Art, science, or both? **The Electronic Alphabet** illustrates that the effective use of fonts is both an art and a science. Studied and applied, effective typography puts power in the word and the work. The right font can convey an attitude, an emotion, the significance of a subject. It can persuade.

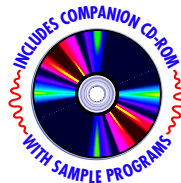
The computer industry gave birth to TrueType fonts, in fact, whole families of fonts. Each as different as fingerprints. Each with its own special font quality. This book helps every communicator understand the history of fonts, their lineage, the "font families", character sets and how they can be used most effectively.

The **Electronic Alphabet** also explains how to design and create your own fonts by using various programs, such as *Fontographer*, *TypeDesigner*, and *CorelDraw!* And if you don't want to design your own, there are numerous sources for fonts, including shareware, which are reviewed in the book.

On the technical side, not scientific, but technical, Windows users will learn how to install and configure fonts in the TrueType, PostScript Type 1, and PCL-V formats. Users will also find tips on solving problems and errors that occur while printing, installing fonts, and deinstalling fonts - some of the realities of working with fonts on computers and with software.

As a special bonus, **Electronic Alphabet** comes with a companion CD-ROM with over 800 fonts, a font manager and a font editor. The font manager has several font viewers, a font archiver, and even lets you print out hard copy of all your fonts. Over 300 MB of font software!

Authors: Bernd Salewski
Order Item: #B259
ISBN: 1-55755-259-2
SRP: \$29.95 US/ \$39.95 CAN with CD-ROM



To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

**Productivity Series books are for users who want
to become more productive with their PC.**

Zippping for Beginners

“No Experience Required”

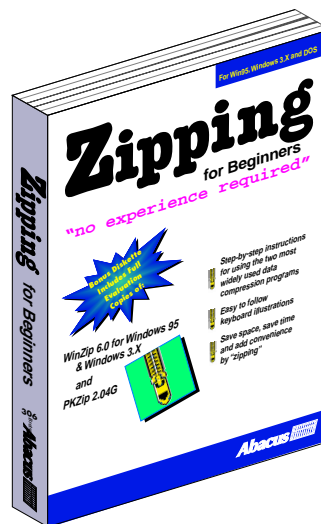
What is zippping?

Zippping is a way to “crunch” files. By zippping, you can save hard disk space, reduce on-line connect time and make it easier for you to pass files onto friends and business associates.

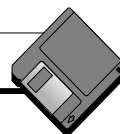
Zippping for Beginners is a no-nonsense introductory guide to using the two most popular zippping utilities – PKZip and WinZip. These step-by-step instructions will turn you into a zippping (and unzipping) expert in just a few short minutes.

Book highlights:

- What’s a “zipped” file?
- Viewing a compressed file
- Zippping several files at once
- Zippping to multiple diskettes
- Creating a self-extracting zip file



Includes ready-to-use
Companion Diskette



Includes a companion diskette with full evaluation copies of WinZip 6.0 for Windows 95 & Windows 3.X and PKZip 2.04G.

Author: Abacus Group

Order Item: #B306

ISBN: 1-55755-306-8

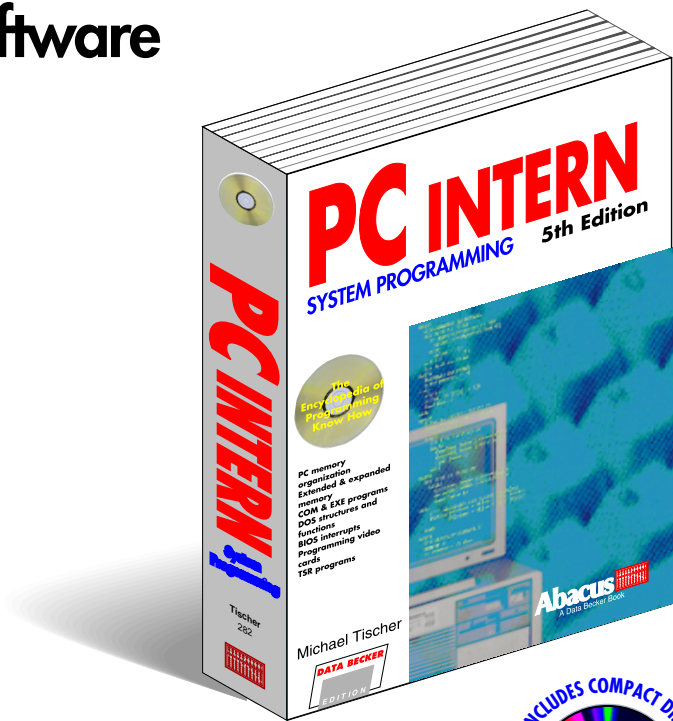
Suggested retail price: \$14.95 US/ \$19.95 CAN with companion diskette

To order direct call Toll Free 1-800-451-4319

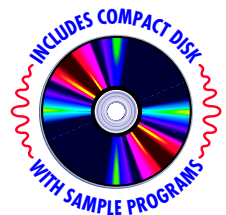
In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

PC catalog

Order Toll Free 1-800-451-4319
Books and Software



Abacus 



 **To order direct call Toll Free 1-800-451-4319**

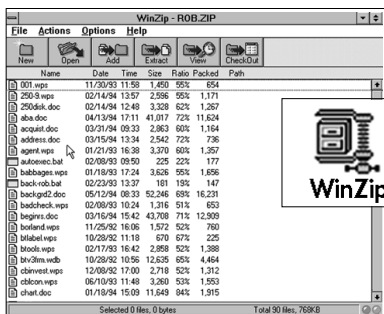
In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Software

ZIP KIT

Learn to Use the World's Most Popular Data Compression Programs!

Start Using WinZip and nine other fully functional evaluation versions of the most popular data compression shareware in the universe today.



PKZip



ARJ



LH-ICE



PKLite



LHArc



LZH

This Windows-based utility makes unlocking the treasures of the information superhighway a breeze. Zip and unzip. Drag-and-drop. Virus scanning support.

Graphic & Video Utilities

Special graphic and video programs in the **ZIP KIT** will help you work with dozens of different formats.

Learn to convert and compress numerous graphic formats with unique image processing software: ART, BMP, DIB, GIF, IFF, ICO, LBM, MSP, PCX, RLE, TIF, WPG, CUT, EXE, HRZ, IMG, JPG, MAC, PIC, RAS, TGA, and TXT.

Discover the world of compressed video files: FLI, AVI, MPG, FLC, and MCI.

ZIP KIT

Item #S287

SRP: \$34.95 US/ \$46.95 CAN

ISBN 1-55755-290-8

UPC 0 90869 55290 1

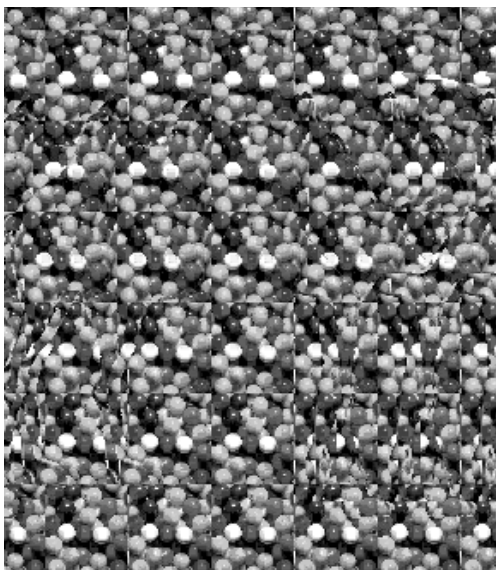
To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Software

STEREOGRAM *Workshop*

Create Your Own 3-D "Hidden" Pictures



The Stereogram Workshop

Stereograms are a worldwide phenomena, spellbinding 3-D art and stereograms can now be created on your own PC! Inside the Stereogram Workshop you'll find all the parts you need in a Windows interface: depth images, foreground images, texture tools, and wallpaper. Create your own calendars, posters and greeting cards with hidden images for your "minds eye". **Now with KidStuff!** Create fun stereograms using any of the dozens of KidStuff animals and textures like gumballs and jellybeans

Stereogram Workshop

Item #S280

ISBN 1-55755-280-0

UPC 0 90869 55280 2

SRP: \$29.95 US/ \$39.95 CAN

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Software

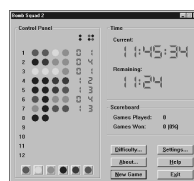
MegaPak for Windows 95

Hundreds of Programs!

- ▶ Internet Tools
- ▶ Windows Utilities
- ▶ Games & Edutainment
- ▶ Animated Icons & Cursors
- ▶ Graphics & Sound

MegaPak for Windows 95
Item #S289
ISBN 1-55755-289-4
UPC 0 90869 55289 5
SRP: \$29.95 US/ \$39.95 CAN

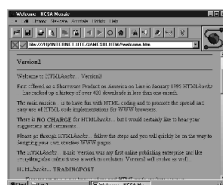
A few of the sample programs -



Bomb Squad



Jukebox



Mosaic

Animated Icons



To order direct call Toll Free 1-800-451-4319

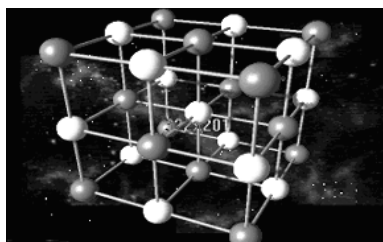
In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Software



CD-ROM! The Naked Truth & Killer Animations

A great, "all in one" package for the CD-ROM, multimedia enthusiast. It's a workshop filled with a ton of software to supercharge your drive and titillate your imagination. Inside you'll walk through the nitty-gritty of CD-ROM technology and into the wildside of virtual landscapes and 3-D animations. Nothing is sacred; everything exposed. The companion book gives you the "naked truth" about CD-ROMs while a CD packed with over 400 MB of software offers the ultimate in utilities and animations.



CD-ROM Workshop
Item #S272
ISBN 1-55755-272-X
UPC 0 90869 55272 7

SRP: \$34.95 US/ \$44.95 CAN

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Software

PHOTO CD

Workshop

The PHOTO CD Workshop is an exciting introduction into the world of photo CD technology, Kodak's replacement for conventional films. This CD-ROM and book bridge the gap between your multimedia system and the latest in digital image technology. Multimedia fans, desktop publishers, and graphic artists can now experiment with photo quality image processing software, morphing effects, and photo CD images. Enjoy the real power of your "photo CD compatible" CD-ROM with the Workshop CD - 400 MB of software.



PHOTO CD Workshop

Item #S262

SRP: \$29.95 US/ \$39.95 CAN

ISBN 1-55755-262-2

UPC 0 90869 55262 8

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Software



FAST-CD is the innovative CD-ROM accelerator for both Windows and DOS. It can boost the access speed of almost any CD-ROM drive by up to 600%.

▶ **FAST-CD** lets you run CD-ROM games, applications or reference works faster without investing in additional hardware

▶ **FAST-CD** uses 'QuickImage' Technology which makes using CD-ROM like working from your hard drive

▶ **FAST-CD's** SpeedCache uses 32-bit memory management for Windows users



▶ Easy installation under DOS & Windows Configurations

▶ Program and test software included on CD

▶ Documentation with detailed installation and configuration examples

▶ Not compatible with compressed hard drives - DoubleSpace, Stacker or DriveSpace

▶ For all widely-used single to quad speed drives

FAST-CD

Item #S281

ISBN 1-55755-281-9

UPC 0 90869 55281 9

SRP: \$34.95 US/ \$44.95 CAN

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

**Productivity Series books are for users who want
to become more productive with their PC.**

Excel 5 Complete, Special Edition

Excel 5 Complete is the latest release in our “Complete” user guide series. Our “Complete” books have earned a reputation among end-users as being thorough, informative, and easy to learn from — for beginners through advanced level computer users — and this book is no exception.

The new version of Microsoft’s Excel, version 5, sets new standards in functionality and user-friendliness. **Excel 5 Complete** covers the full functional spectrum of this widely used spreadsheet program, with special emphasis on the new features in version 5.

Beginners and users upgrading from previous versions of Excel will learn from the introductory chapter in which new features, fundamentals, and special aspects of Excel 5 are introduced in an easy-to-understand manner. From there, the user is led, step-by-step, into creating a spreadsheet, the use of the new pivot views, working with formulas, and the diverse opportunities for using data exchange.

A special concentration is on the graphical representation of values in the form of tables or complete presentations (including multimedia). The new database functions are also introduced, as are the fantastic options available with object-oriented data exchange through OLE 2.0 (object linking and embedding).

Other subjects covered include:

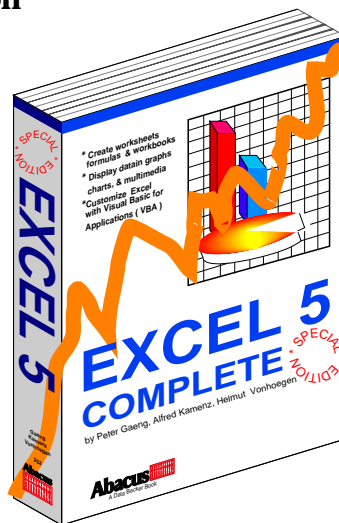
- ◆ Using macros
- ◆ Introduction to VBA programming language (Visual Basic for Applications)
- ◆ Example oriented function reference.
- ◆ Companion diskettes with all of the example applications from the book

Authors: Peter Gaeng, Alfred Kamenz & Helmut Vonhoegen

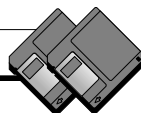
Order Item: #B252

ISBN: 1-55755-252-5

Suggested retail price: \$34.95 US / \$44.95 CAN with companion diskettes



Includes ready-to-use
Companion Diskettes



To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

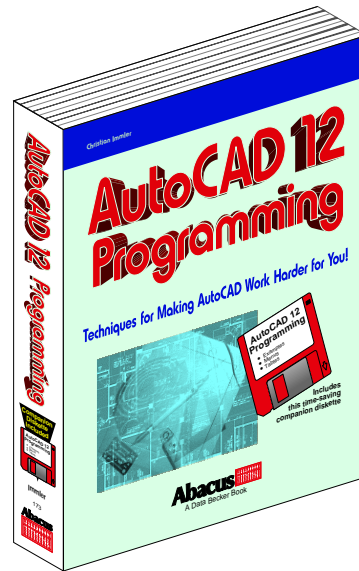
Developers Series books are for professional software developers who require in-depth technical information and programming techniques.

AutoCAD 12 Programming

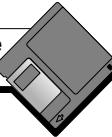
AutoCAD 12 Programming will teach you how to integrate special custom functions and commands into your AutoCAD system. **AutoCAD 12 Programming** presents the many options of AutoCAD programming in a well-founded, yet easy to understand format. This guide includes thorough descriptions of how to create custom work environments and custom menus. **AutoCAD 12 Programming** explains BATCH programming (for user defined startup of Auto CAD 12), creating Script files (for specific drawing sequences) and programming custom commands with AutoLISP or ADS.

Also includes:

- Overview of AutoCAD 12
- Installation tips
- Prototype drawing and output functions
- Script programming
- AutoLISP, menus and commands
- AutoCAD Development System
- Practical information to adapt AutoCAD to your requirements.



**Includes ready-to-use
Companion Diskette**



AutoCAD 12 Programming includes a companion disk that contains programming examples and ready to use menus.

Author: Christian Immmler

ISBN: 1-55755-173-1

Order Item: #B173

Suggested retail price: \$44.95 US/ \$54.95 CAN with companion diskette

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Productivity Series books are for users who want to become more productive with their PC.

Warping To The Internet

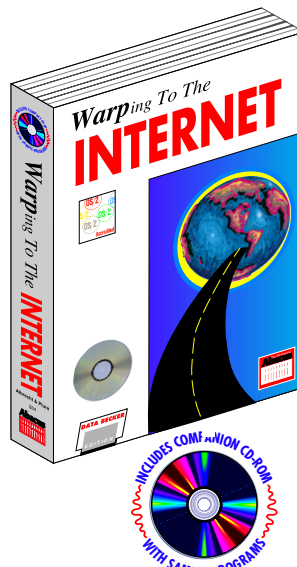
The Internet grows by leaps and bounds every day. Feedback from first time users ranges from "easy" to an "absolute nightmare". This book shows you how easy it can be to access and benefit from the Internet using OS/2 Warp. From installation to tuning tips to on-line sessions, **Warping to the Internet** is a practical guide to getting connected to the Internet with OS/2. It takes the mystery out of the information highway.

Warping to the Internet introduces users to the suite of Internet applications bundled into OS/2's easy-to-use graphical shell. From the now famous Gopher and Internet E-Mail to the quick access of global newsgroups, Warp users will learn how easy it is to log on and navigate the world.

By accessing the Internet, you're able to connect to any of thousands of computer sites worldwide which can instantly deliver information to you. But knowing how to find these sites and how to locate the desired specific information is the big challenge. **Warping to the Internet** shows you how to sort your way through the Internet and find the information you need. Whether you're solving a business problem, researching a school assignment, investigating a medical issue or tinkering at your hobby, **Warping to the Internet** will help you get there.

"Like the Library of Congress without a card catalog" - that's what it can be like to search the Internet. To help users conserve precious connect-time while on the Internet, the companion CD-ROM is filled with easy to search lists and directories of information. This CD-ROM is a resource that can help you quickly navigate to the correct destination before logging on to the Internet.

Authors: Norbert Salomon
Order Item: #B284
ISBN: 1-55755-284-3
SRP: \$29.95 US/ \$39.95 CAN with CD-ROM



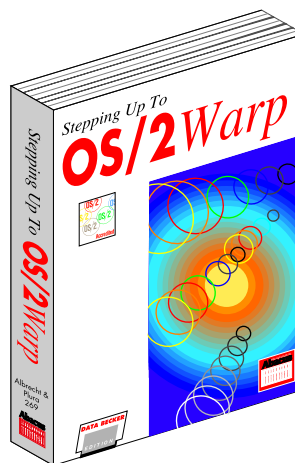
To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

**Productivity Series books are for users who want
to become more productive with their PC.**

Stepping Up To OS/2 Warp

Stepping Up To OS/2 Warp is a fast paced way to move to OS/2 - from installation and configuration to your first encounter with the Workplace Shell. You'll also see how to configure and use your DOS and Windows applications, take advantage of the multitasking features, adjust the look and feel of OS/2 objects using notebook settings, Dual Boot procedures, and install and use the BonusPak applications including the IBM Internet Connection and CompuServe Information Manager. In short, it's the fastest way to get up and running with Warp.



Covers these essential skills:

- Installing OS/2 Warp - using the new Easy Installation
- Understanding the High Performance File Systems - HPFS
- Working alternately with two operating systems - Dual Boot
- Using programs of the Productivity Folder
- Using the Workplace Shell
- Working in DOS Windows
- Multitasking and data exchange
- Using Windows programs
- Complete command reference
- BonusPak information
- Using Photo CD images with Multimedia Viewer

Authors: Albrecht & Plura

Order Item: #B269

ISBN: 1-55755-269-X

SRP: \$19.95 US/ \$26.95 CAN

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

Software

GigaPak *for OS/2 Warp*

Two Super CD-ROMs **Hundreds of OS/2 Programs**

- ▶ **Presentation Manager Tools**
- ▶ **OS/2 Utilities**
- ▶ **Games & Entertainment**
- ▶ **Icons**
- ▶ **Multimedia**
- ▶ **Graphics & Music**
- ▶ **Device Drivers**
- ▶ **OS/2 Programming Tools**

GigaPak for OS/2 Warp
Item #S286
ISBN 1-55755-286-X
UPC 0 90869 55286 4
SRP: \$29.95 US/ \$39.95 CAN

Here are just a few samples



PM Calculator



Tetris



Jukebox



PM Zipper



PM Draw



PM Backup

To order direct call Toll Free 1-800-451-4319

In US and Canada add \$5.00 shipping and handling. Foreign orders add \$13.00 per item.
Michigan residents add 6% sales tax.

ORDER FORM

First Name: | | | | | | | | | | | | | | | | | |

Last Name: _____

Company (If Applicable):_____

Address: _____

City: _____ **State:** _____ **Zip:** _____

Country: _____

Phone: _____ **Fax:** _____

For immediate delivery - Order Toll Free 1-800-451-4319

Prices and availability subject to change

Qty	Item	Title	Price each	Ext. Price
Method of payment: <input type="checkbox"/> Visa <input type="checkbox"/> Am Express <input type="checkbox"/> Mastercard <input type="checkbox"/> Check/ M.O.			Subtotal	
		Sales Tax:	Michigan Residents add 6% sales tax	
		Shipping (Choose one)	US and Canada add \$5.00 per order	
			Or Foreign orders \$13.00 per item	
			TOTAL amount enclosed (US funds)	

Card No. _____ Exp. Date _____

Signature _____

Abacus

5370 52nd Street SE • Grand Rapids, MI 49512
Phone (616) 698-0330 • FAX (616) 698-0325